

Docket No.: 57454-335

*112*  
PATENT

JC978 U.S. PRO  
10/073215  
02/13/02

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of :

Masato HAGIWARA, et al. :

Serial No.: :

Group Art Unit:

Filed: February 13, 2002 :

Examiner:

For: PROGRAM EXECUTION DEVICE OPERATING BASED ON COMPRESSED CODE

**CLAIM OF PRIORITY AND  
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Commissioner for Patents  
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

**Japanese Patent Application No. 2001-124824(P), filed April 23, 2001**

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

*Becker*  
Stephen A. Becker  
Registration No. 26,527

600 13<sup>th</sup> Street, N.W.  
Washington, DC 20005-3096  
(202) 756-8000 SAB:mlw  
**Date: February 13, 2002**  
Facsimile: (202) 756-8087

57454-335  
Masato HAGIWARA et al.  
February 13, 2002

日 本 国 特 許 庁

JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 4月23日

出 願 番 号

Application Number:

特願2001-124824

出 願 人

Applicant(s):

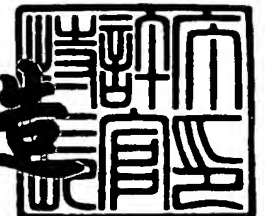
三菱電機株式会社



2001年 5月18日

特 許 庁 長 官  
Commissioner,  
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3042668

【書類名】 特許願

【整理番号】 530049JP01

【提出日】 平成13年 4月23日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/45

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会  
社内

【氏名】 萩原 正人

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会  
社内

【氏名】 吉田 豊彦

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会  
社内

【氏名】 坂本 守

【特許出願人】

【識別番号】 000006013

【氏名又は名称】 三菱電機株式会社

【代理人】

【識別番号】 100064746

【弁理士】

【氏名又は名称】 深見 久郎

【選任した代理人】

【識別番号】 100085132

【弁理士】

【氏名又は名称】 森田 俊雄

【選任した代理人】

【識別番号】 100091409

【弁理士】

【氏名又は名称】 伊藤 英彦

【選任した代理人】

【識別番号】 100096781

【弁理士】

【氏名又は名称】 堀井 豊

【選任した代理人】

【識別番号】 100096792

【弁理士】

【氏名又は名称】 森下 八郎

【手数料の表示】

【予納台帳番号】 008693

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要



【書類名】 明細書

【発明の名称】 プログラム実行装置および方法

【特許請求の範囲】

【請求項 1】 所定の言語で記述されたプログラムを実行するプログラム実行装置であって、

前記プログラムの、所定の単位毎に圧縮されたコードを格納する圧縮コード格納部と、

前記圧縮コード格納部に接続され、前記圧縮コード格納部から前記圧縮されたコードを読み出して伸張するための伸張手段と、

前記伸張手段に接続され、前記伸張手段により伸張されたコードを格納するコード格納部と、

前記コード格納部に接続され、伸張された前記コードを解釈し、実行するためのインタープリタ手段とを含む、プログラム実行装置。

【請求項 2】 前記所定の言語はオブジェクト指向言語であり、前記所定の単位は、メソッドである、請求項 1 に記載のプログラム実行装置。

【請求項 3】 前記所定の単位は、プログラム中で分岐を含まない一連の命令である、請求項 1 に記載のプログラム実行装置。

【請求項 4】 前記所定の単位は、命令である、請求項 1 に記載のプログラム実行装置。

【請求項 5】 さらに、前記圧縮コード格納部に接続され、圧縮された前記コードに基づいて、前記コードの圧縮方式を判定するための圧縮方式判定手段を含み、

前記伸張手段は、前記圧縮コード格納部、前記コード格納部および前記圧縮方式判定手段に接続され、前記圧縮方式判定手段の出力に基づいて、圧縮された前記コードを伸張し、前記コード格納部に格納するための手段を含む、請求項 1 に記載のプログラム実行装置。

【請求項 6】 オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置であって、前記プログラムは前記プログラム実行装置にとってネイティブなコード以外のコードで記述されており、

メソッド毎に圧縮された前記プログラムのコードを格納する圧縮コード格納部と、

前記圧縮コード格納部に接続され、圧縮された前記コードを伸張するための伸張手段と、

前記伸張手段に接続され、伸張された前記コードをネイティブコードに変換するための変換手段と、

前記変換手段に接続され、前記変換手段により出力されるネイティブコードを格納するネイティブコード格納部と、

前記ネイティブコード格納部に接続され、前記ネイティブコードを実行するためのネイティブコード実行手段とを含む、プログラム実行装置。

【請求項 7】 前記ネイティブコード格納部は、キャッシュメモリよりなる、請求項 6 に記載のプログラム実行装置。

【請求項 8】 オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置であって、

前記プログラムのメソッドのコードを格納するコード格納部と、

メソッドのネイティブコードを格納するネイティブコード格納部と、

メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部と、

前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するための第 1 の判断手段と、

前記圧縮ネイティブコード格納部に接続され、前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断する第 2 の判断手段と、

前記第 1 および第 2 の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部および前記ネイティブコード格納部に接続され、前記第 1 および第 2 の判断手段の出力に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティ

ブコード格納部へ格納するための、ネイティブコード格納制御手段と、

前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に格納されたネイティブコードを実行するためのネイティブコード実行手段と、

前記第 2 の判断手段、前記ネイティブコード格納部および前記圧縮ネイティブコード格納部に接続され、前記第 2 の判断手段の出力に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するためのネイティブコード圧縮格納手段とを含む、プログラム実行装置。

【請求項 9】 さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含み、

前記ネイティブコード格納制御手段は、前記第 1 および第 2 の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部、前記ネイティブコード格納部および前記圧縮方式格納部に接続され、前記第 1 および第 2 の判断手段の出力に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するための手段を含み、

前記ネイティブコード圧縮格納手段は、前記第 2 の判断手段、前記ネイティブコード格納部、前記圧縮ネイティブコード格納部および前記圧縮方式格納部に接続され、前記第 2 の判断手段の出力に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するための手段を含む、請求項 8 に記載のプログラム実行装置。

【請求項 10】 前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する、請求項 8 に記載のプログラム実行装置。

【請求項 11】 前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低

いものから優先的に圧縮する、請求項 8 に記載のプログラム実行装置。

【請求項 1 2】 前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する、請求項 8 に記載のプログラム実行装置。

【請求項 1 3】 前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が大きいものから優先的に圧縮する、請求項 8 に記載のプログラム実行装置。

【請求項 1 4】 前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する、請求項 8 に記載のプログラム実行装置。

【請求項 1 5】 前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する、請求項 8 に記載のプログラム実行装置。

【請求項 1 6】 前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する、請求項 8 に記載のプログラム実行装置。

【請求項 1 7】 前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する、請求項 8 に記載のプログラム実行装置。

【請求項 1 8】 オブジェクト指向言語で記述されたプログラムを実行するプログラム実行方法であって、

所定の単位毎に圧縮されたコードを伸張するステップと、

伸張されたコードを解釈し、実行するステップとを含む、プログラム実行方法

。

【請求項 1 9】 前記所定の単位は、メソッドである、請求項 1 8 に記載の

プログラム実行方法。

【請求項 2 0】 前記所定の単位は、プログラム中で分岐を含まない一連の命令である、請求項 1 8 に記載のプログラム実行方法。

【請求項 2 1】 前記所定の単位は、命令である、請求項 1 8 に記載のプログラム実行方法。

【請求項 2 2】 さらに、所定の単位毎に圧縮されたコードの圧縮方式を判定するステップを含み、

前記伸張するステップは、判定された圧縮方式に基づいて、所定の単位毎に圧縮されたコードを伸張するステップを含む、請求項 1 8 に記載のプログラム実行方法。

【請求項 2 3】 オブジェクト指向言語で記述されたプログラムを実行するプログラム実行方法であって、

メソッド毎に圧縮されたコードを伸張するステップと、

伸張された前記コードをネイティブコードに変換するステップと、

前記ネイティブコードを実行するステップとを含む、プログラム実行方法。

【請求項 2 4】 オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置で用いられるプログラム実行方法であって、

前記プログラム実行装置は、前記プログラムのメソッドのコードを格納するコード格納部と、メソッドのネイティブコードを格納するネイティブコード格納部と、メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部とを含み、

前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するステップと、

前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断するステップと、

前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部

へ格納するステップと、

前記ネイティブコード格納部に格納されたネイティブコードを実行するステップと、

前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するステップとを含む、プログラム実行方法。

【請求項 2 5】 前記プログラム実行装置は、さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含み、

前記ネイティブコードを前記ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するステップを含み、

前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するステップを含む、請求項 2 4 に記載のプログラム実行方法。

【請求項 2 6】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する、請求項 2 4 に記載のプログラム実行方法。

【請求項 2 7】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低いものから優先的に圧縮する、請求項 2 4 に記載のプログラム実行方法。

【請求項 2 8】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する、請求項 2 4 に記載のプログラム実行方法。

【請求項 2 9】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が大きいものから優先的に圧縮する、請求項 2 4 に記載のプログラム実行方法。

【請求項 3 0】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する、請求項 2 4 に記載のプログラム実行方法。

【請求項 3 1】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する、請求項 2 4 に記載のプログラム実行方法。

【請求項 3 2】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する、請求項 2 4 に記載のプログラム実行方法。

【請求項 3 3】 前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する、請求項 2 4 に記載のプログラム実行方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、プログラム実行装置および方法に関し、特に、データを格納するメモリの容量が小さいプログラム実行装置および方法に関する。

【0002】

【従来の技術】

J a v a (R) (米国サンマイクロシステムズ社の登録商標) 言語で記述されたプログラムは、実行に先だってバイトコードと呼ばれるプラットフォームに依存しないコードにコンパイルされ、配布される。J a v a (R) 仮想マシンは、インタプリタを用いてバイトコードを1命令ずつ解釈しながら実行することにより、プログラムの実行を行なう。最近では、バイトコードをそのまま実行するのではなく、J I T (Just-in-time Compiler) を用いて、バイトコードをネイティブコードに変換し、プログラムを高速に実行できるようにしたものもある。

【0003】

このような、J a v a (R) 言語で記述されたプログラムは、携帯電話機、PDA (Personal Digital Assistants) または情報家電などの組込み機器においても広く用いられている。

【0004】

【発明が解決しようとする課題】

しかし、組込み機器では、使用できるROM (Read Only Memory) またはRAM (Random Access Memory) の容量が通常のコンピュータに比べて低い。このため、ステップ数の大きなプログラムを実行できないという問題がある。

【0005】

本発明は上述の課題を解決するためになされたもので、その目的は、必要なメモリの容量が小さいプログラム実行装置および方法を提供することである。

【0006】

【課題を解決するための手段】

本発明のある局面に従うプログラム実行装置は、所定の言語で記述されたプログラムを実行する。プログラム実行装置は、前記プログラムの、所定の単位毎に



圧縮されたコードを格納する圧縮コード格納部と、前記圧縮コード格納部に接続され、前記圧縮コード格納部から前記圧縮されたコードを読み出して伸張するための伸張手段と、前記伸張手段に接続され、前記伸張手段により伸張されたコードを格納するコード格納部と、前記コード格納部に接続され、伸張された前記コードを解釈し、実行するためのインタープリタ手段とを含む。

## 【0007】

所定の単位毎にコードが圧縮されて圧縮コード格納部に格納されている。このため、圧縮コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

## 【0008】

好ましくは、前記所定の言語はオブジェクト指向言語であり、前記所定の単位は、メソッドである。

## 【0009】

さらに好ましくは、前記所定の単位は、プログラム中で分岐を含まない一連の命令である。

## 【0010】

コード格納部には、分岐を含まない一連の命令のコードが格納される。このため、メソッドのコードを格納する場合に比べ、コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

## 【0011】

さらに好ましくは、前記所定の単位は、命令である。

コード格納部には、命令のコードが格納される。このため、メソッドのコードを格納する場合に比べ、コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

## 【0012】

さらに好ましくは、プログラム実行装置は、さらに、前記圧縮コード格納部に接続され、圧縮された前記コードに基づいて、前記コードの圧縮方式を判定するための圧縮方式判定手段を含む。前記伸張手段は、前記圧縮コード格納部、前記

コード格納部および前記圧縮方式判定手段に接続され、前記圧縮方式判定手段の出力に基づいて、圧縮された前記コードを伸張し、前記コード格納部に格納するための手段を含む。

【0013】

メソッド毎にコードの圧縮方式をユーザが選択することができ、最適な圧縮方式を採用することができる。このため、圧縮コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【0014】

本発明の他の局面に従うプログラム実行装置は、オブジェクト指向言語で記述されたプログラムを実行する。前記プログラムは前記プログラム実行装置にとってネイティブなコード以外のコードで記述されている。プログラム実行装置は、メソッド毎に圧縮された前記プログラムのコードを格納する圧縮コード格納部と、前記圧縮コード格納部に接続され、圧縮された前記コードを伸張するための伸張手段と、前記伸張手段に接続され、伸張された前記コードをネイティブコードに変換するための変換手段と、前記変換手段に接続され、前記変換手段により出力されるネイティブコードを格納するネイティブコード格納部と、前記ネイティブコード格納部に接続され、前記ネイティブコードを実行するためのネイティブコード実行手段とを含む。

【0015】

メソッドのコードが圧縮されて圧縮コード格納部に格納されている。このため、圧縮コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【0016】

好ましくは、前記ネイティブコード格納部は、キャッシュメモリよりなる。

ネイティブコードをキャッシュメモリ内に格納することにより、プログラムの実行速度を向上させることができる。

【0017】

本発明のさらに他の局面に従うプログラム実行装置は、オブジェクト指向言語

で記述されたプログラムを実行する。プログラム実行装置は、前記プログラムのメソッドのコードを格納するコード格納部と、メソッドのネイティブコードを格納するネイティブコード格納部と、メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部と、前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するための第1の判断手段と、前記圧縮ネイティブコード格納部に接続され、前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断する第2の判断手段と、前記第1および第2の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部および前記ネイティブコード格納部に接続され、前記第1および第2の判断手段の出力に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するための、ネイティブコード格納制御手段と、前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に格納されたネイティブコードを実行するためのネイティブコード実行手段と、前記第2の判断手段、前記ネイティブコード格納部および前記圧縮ネイティブコード格納部に接続され、前記第2の判断手段の出力に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するためのネイティブコード圧縮格納手段とを含む。

## 【 0 0 1 8 】

コードがネイティブコードに変換された後、圧縮されて圧縮ネイティブコード格納部に格納される。このため、JITを用いた従来の装置に比べ、ネイティブコード格納部の容量を小さくすることができる。

## 【 0 0 1 9 】

好ましくは、プログラム実行装置は、さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含む。前記ネイティブコード格納制御手段は、前記第1および第2の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部、前記ネイ

ティブコード格納部および前記圧縮方式格納部に接続され、前記第 1 および第 2 の判断手段の出力に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するための手段を含む。前記ネイティブコード圧縮格納手段は、前記第 2 の判断手段、前記ネイティブコード格納部、前記圧縮ネイティブコード格納部および前記圧縮方式格納部に接続され、前記第 2 の判断手段の出力に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するための手段を含む。

## 【 0 0 2 0 】

メソッド毎に最適な圧縮方式を用いてネイティブコードの圧縮が行なわれる。このため、圧縮ネイティブコード格納部の容量を小さくすることができる。

## 【 0 0 2 1 】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する。

## 【 0 0 2 2 】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低いものから優先的に圧縮する。

## 【 0 0 2 3 】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する。

## 【 0 0 2 4 】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が

大きいものから優先的に圧縮する。

【0025】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する。

【0026】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する。

【0027】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する。

【0028】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する。

【0029】

本発明のさらに他の局面に従うプログラム実行方法は、オブジェクト指向言語で記述されたプログラムを実行する。プログラム実行方法は、所定の単位毎に圧縮されたコードを伸張するステップと、伸張されたコードを解釈し、実行するステップとを含む。

【0030】

圧縮されたコードを伸張してから、コードの解釈および実行が行われる。このようにコードは予め圧縮されているため、小さなメモリ容量の装置においても、

プログラムを実行することができる。

【0031】

好ましくは、前記所定の単位は、メソッドである。

さらに好ましくは、前記所定の単位は、プログラム中で分岐を含まない一連の命令である。

【0032】

分岐を含まない一連の命令単位でコードが圧縮される。このため、メソッド単位でコードが圧縮される場合に比べ、伸張されたコードを記憶するメモリの容量が少なくすむ。

【0033】

さらに好ましくは、前記所定の単位は、命令である。

命令単位でコードが圧縮される。このため、メソッド単位でコードが圧縮される場合に比べ、伸張されたコードを記憶するメモリの容量が少なくすむ。

【0034】

さらに好ましくは、プログラム実行方法は、さらに、所定の単位毎に圧縮されたコードの圧縮方式を判定するステップを含む。前記伸張するステップは、判定された圧縮方式に基づいて、所定の単位毎に圧縮されたコードを伸張するステップを含む。

【0035】

メソッド毎にコードの圧縮方式をユーザが選択することができ、最適な圧縮方式を採用することができる。このため、圧縮されたコードを格納するメモリの容量を小さくすることができる。

【0036】

本発明のさらに他の局面に従うプログラム実行方法は、オブジェクト指向言語で記述されたプログラムを実行する。プログラム実行方法は、メソッド毎に圧縮されたコードを伸張するステップと、伸張された前記コードをネイティブコードに変換するステップと、前記ネイティブコードを実行するステップとを含む。

【0037】

メソッドのコードが圧縮されている。このため、メソッドのコードを格納する

メモリの容量を小さくすることができる。

【0038】

本発明のさらに他の局面に従うプログラム実行方法は、オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置で用いられる。前記プログラム実行装置は、前記プログラムのメソッドのコードを格納するコード格納部と、メソッドのネイティブコードを格納するネイティブコード格納部と、メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部とを含む。プログラム実行方法は、前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するステップと、前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断するステップと、前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するステップと、前記ネイティブコード格納部に格納されたネイティブコードを実行するステップと、前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するステップとを含む。

【0039】

コードがネイティブコードに変換された後、圧縮されて圧縮ネイティブコード格納部に格納される。このため、JITを用いた従来の装置に比べ、ネイティブコード格納部の容量を小さくすることができる。

【0040】

好ましくは、前記プログラム実行装置は、さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含む。前記ネイティブコードを前記ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコード

の伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するステップを含む。前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するステップを含む。

【0041】

メソッド毎に最適な圧縮方式を用いてネイティブコードの圧縮が行なわれる。このため、圧縮ネイティブコード格納部の容量を小さくすることができる。

【0042】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する。

【0043】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低いものから優先的に圧縮する。

【0044】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する。

【0045】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が大きいものから優先的に圧縮する。

【0046】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティ



ブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する。

## 【0047】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する。

## 【0048】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する。

## 【0049】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する。

## 【0050】

## 【発明の実施の形態】

## 〔実施の形態1〕

本発明の第1の実施の形態に係るJava(R)仮想マシンは、組込み機器を用いて実現される。

## 【0051】

図1を参照して、組込み機器は、プログラムを解釈、実行するCPU(Central Processing Unit)2と、CPU2で実行されるプログラムのバイトコードを記憶するROM(Read Only Memory)4と、そのプログラムを実行するための各種データを記憶するRAM(Random Access Memory)8と、ユーザとの間でデータの入出力を行なうためのユーザインタフェース6と、CPU2、ROM4、ユーザインタフェース6およびRAM8を相互に接続するバスを含む。

## 【 0 0 5 2 】

図 2 を参照して、ROM 4 は、J a v a ( R ) 言語で記述されたプログラムのバイトコードを圧縮したもの（以下「圧縮バイトコード」と言う。）を格納する圧縮バイトコード格納領域 1 2 と、その他のクラス情報を格納する領域 1 4 とを含む。

## 【 0 0 5 3 】

図 3 を参照して、RAM 8 は、オペレーティングシステムを格納するオペレーティングシステム格納領域 1 6 と、バーチャルマシン（VM）モジュールを格納する VM モジュール格納領域 1 8 と、データを圧縮するためのプログラムである圧縮モジュールを格納するための圧縮モジュール格納領域 2 4 と、圧縮されたデータを伸張するためのプログラムである伸張モジュールを格納するための伸張モジュール格納領域 2 6 と、バイトコードおよびネイティブコード等を格納するためのコード格納領域 2 8 と、メソッドの状態を表わすメソッドステータスを格納するためのメソッドステータス格納領域 3 6 とを含む。メソッドステータス格納領域 3 6 はメソッド毎に設けられる。

## 【 0 0 5 4 】

VM モジュール格納領域 1 8 は、インタプリタを格納するためのインタプリタ格納領域 2 0 と、J I T を格納するための J I T 格納領域 2 2 とを含む。

## 【 0 0 5 5 】

コード格納領域 2 8 は、伸張済みバイトコードを格納する伸張済みバイトコード格納領域 3 0 と、ネイティブコードを格納するネイティブコード格納領域 3 2 と、ネイティブコードを圧縮したもの（以下「圧縮ネイティブコード」と言う。）を格納する圧縮ネイティブコード格納領域 3 4 とを含む。

## 【 0 0 5 6 】

メソッドステータス格納領域 3 6 は、メソッドが実行された回数を格納する回数格納領域 3 8 と、メソッドのサイズを格納するサイズ格納領域 4 0 と、ネイティブコードが存在するか否かを示すネイティブコード有無フラグを格納するネイティブコード有無フラグ格納領域 4 2 と、バイトコードがコンパイルされた時刻または順番を格納するコンパイル時刻（順番）格納領域 4 4 と、圧縮に関する情

報を格納する圧縮情報格納領域 4 6 とを含む。

【 0 0 5 7 】

圧縮情報格納領域 4 6 は、メソッドが圧縮されているか否かを示す圧縮フラグを格納する圧縮フラグ格納領域 4 8 と、圧縮されたメソッドが伸張されているか否かを示す伸張済みフラグを格納する伸張済みフラグ格納領域 5 0 と、メソッドの圧縮方式を格納する圧縮方式格納領域 5 2 と、圧縮の際の圧縮率を格納する圧縮率格納領域 5 4 と、圧縮された時刻または順番を格納する圧縮時刻（順番）格納領域 5 6 と、圧縮ネイティブコードのサイズを格納する圧縮サイズ格納領域 5 8 とを含む。

【 0 0 5 8 】

図 4 を参照して、本実施の形態では、J a v a ( R ) 言語で記述されたプログラムは、メソッド 1 ~ 4 の 4 つのメソッドより構成されるものとして説明を行なう。プログラムがそれ以外の個数のメソッドより構成されている場合であっても、同様の動作を行なう。

【 0 0 5 9 】

すべてのメソッドは、メソッド毎に圧縮され、圧縮バイトコード格納領域 1 2 に格納されている。

【 0 0 6 0 】

図 5 を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、C P U 2 は、呼出されたメソッドの伸張済みフラグ格納領域 5 0 に格納されている伸張済みフラグを参照し、呼出されたメソッドのバイトコードが、伸張されており、伸張済みバイトコード格納領域 3 0 に格納されているか否かを調べる（S 2）。

【 0 0 6 1 】

バイトコードが伸張されていなければ（S 2 で N O）、圧縮バイトコード格納領域 1 2 に格納されたバイトコードが伸張され（S 4）、伸張後のバイトコードが伸張済みバイトコード格納領域 3 0 に格納される（S 6）。次に、このメソッドの圧縮バイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（S 8）。

## 【 0 0 6 2 】

S 8 の処理の後、またはメソッドのバイトコードが伸張済みバイトコード格納領域 3 0 に格納されている場合には（S 2 で Y E S）、C P U 2 はインタプリタを用いて、伸張済みのバイトコードを 1 命令ずつ解釈しながら実行する（S 1 0）。その後、呼出し側のモジュールへ復帰するための処理を行なう（S 1 2）。

## 【 0 0 6 3 】

なお、伸張済みバイトコード格納領域 3 0 に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応したメソッドの伸張済みフラグがオフに設定される。

## 【 0 0 6 4 】

上述の 4 つのメソッドがメソッド 1、メソッド 2、メソッド 3、メソッド 1、メソッド 2、メソッド 4、メソッド 4、メソッド 4 の順に呼出されて、実行される場合を想定する。また、伸張済みバイトコード格納領域 3 0 には、1 つ分のメソッドのバイトコードしか格納できないとする。このとき、図 6 を参照して、C P U 2 は以下の順序で処理を実行する。

## 【 0 0 6 5 】

メソッド 1 の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（S 2 2）。メソッド 1 のバイトコードを 1 命令ずつ解釈しながら実行する（S 2 4）。メソッド 2 の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（S 2 6）。メソッド 2 のバイトコードを 1 命令ずつ解釈しながら実行する（S 2 8）。メソッド 3 の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（S 3 0）。メソッド 3 のバイトコードを 1 命令ずつ解釈しながら実行する（S 3 2）。メソッド 1 の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（S 3 4）。メソッド 1 のバイトコードを 1 命令ずつ解釈しながら実行する（S 3 6）。

## 【 0 0 6 6 】

メソッド 2 の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（S 3 8）。メソッド 2 のバイトコードを 1 命令ずつ解釈しながら実行する（S 4 0）。メソッド 4 の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（S 4

2)。メソッド4のバイトコードを1命令ずつ解釈しながら実行する（S44）。2つめのメソッド4は伸張済みであるため、メソッド4のバイトコードを1命令ずつ解釈しながら実行する（S46）。3つめのメソッド4は伸張済みであるため、メソッド4のバイトコードを1命令ずつ解釈しながら実行する（S48）。

【0067】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてROMに記憶されている。このため、ROMの容量を小さくすることができる。

【0068】

〔実施の形態2〕

本実施の形態は、第1の実施の形態と異なり、圧縮するバイトコードの単位がメソッド単位ではなく基本ブロック単位である。基本ブロックとは、その中に分岐命令を含まない一連の命令のことをいう。

【0069】

本実施の形態に係るJava（R）仮想マシンは、図1を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0070】

また、ROM4に記憶されている情報は図2を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0071】

なお、圧縮バイトコード格納領域12には、基本ブロック単位でバイトコードが圧縮され、記憶されているものとする。

【0072】

図7を参照して、RAM8は、オペレーティングシステムを格納するオペレーティングシステム格納領域16と、バーチャルマシン（VM）モジュールを格納するVMモジュール格納領域18と、データの圧縮するためのプログラムである圧縮モジュールを格納するための圧縮モジュール格納領域24と、圧縮されたデータを伸張するためのプログラムである伸張モジュールを格納するための伸張モ

ジュール格納領域 2 6 と、バイトコードおよびネイティブコード等を格納するためのコード格納領域 2 8 と、基本ブロックの状態を表わす基本ブロックステータスを格納するための基本ブロックステータス格納領域 6 0 とを含む。基本ブロックステータス格納領域 6 0 は基本ブロック毎に設けられる。

## 【 0 0 7 3 】

基本ブロックステータス格納領域 6 0 は、基本ブロックが実行された回数を格納する回数格納領域 6 2 と、基本ブロックのサイズを格納するサイズ格納領域 6 4 と、ネイティブコードが存在するか否かを示すネイティブコード有無フラグを格納するネイティブコード有無フラグ格納領域 6 6 と、バイトコードがコンパイルされた時刻または順番を格納するコンパイル時刻（順番）格納領域 6 8 と、圧縮に関する情報を格納する圧縮情報格納領域 7 0 とを含む。

## 【 0 0 7 4 】

圧縮情報格納領域 7 0 は、基本ブロックが圧縮されているか否かを示す圧縮フラグを格納する圧縮フラグ格納領域 7 2 と、圧縮された基本ブロックが伸張されているか否かを示す伸張済みフラグを格納する伸張済みフラグ格納領域 7 4 と、基本ブロックの圧縮方式を格納する圧縮方式格納領域 7 6 と、圧縮の際の圧縮率を格納する圧縮率格納領域 7 8 と、圧縮された時刻または順番を格納する圧縮時刻（順番）格納領域 8 0 と、圧縮ネイティブコードのサイズを格納する圧縮サイズ格納領域 8 2 とを含む。

## 【 0 0 7 5 】

図 8 を参照して、上位のモジュールよりある基本ブロックの呼出しがあった場合には、CPU 2 は、呼出された基本ブロックの伸張済みフラグ格納領域 7 4 に格納されている伸張済みフラグを参照し、呼出された基本ブロックのバイトコードが伸張されており、伸張済みバイトコード格納領域 3 0 に格納されているか否かを調べる（S 5 2）。

## 【 0 0 7 6 】

バイトコードが伸張されていなければ（S 5 2 で NO）、圧縮バイトコード格納領域 1 2 に格納されたバイトコードが伸張され（S 5 4）、伸張後のバイトコードが伸張済みバイトコード格納領域 3 0 に格納される（S 5 6）。次に、この

基本ブロックのバイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（S 5 8）。

【0 0 7 7】

S 5 8 の処理の後、または基本ブロックのバイトコードが伸張済みバイトコード格納領域 3 0 に格納されている場合には（S 5 2 で Y E S）、C P U 2 はインタプリタを用いて、伸張済みのバイトコードを 1 命令ずつ解釈しながら実行する（S 6 0）。その後、呼出し側のモジュールへ復帰するための処理を行なう（S 6 2）。

【0 0 7 8】

なお、伸張済みバイトコード格納領域 3 0 に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応した基本ブロックの伸張済みフラグがオフに設定される。

【0 0 7 9】

以上説明したように、本実施の形態によると、バイトコードが圧縮されて R O M に記憶されている。このため、R O M の容量を小さくすることができる。

【0 0 8 0】

また、伸張済みバイトコード格納領域には、基本ブロックの圧縮バイトコードを伸張したバイトコードが格納される。このため、第 1 の実施の形態に比べて伸張済みバイトコード格納領域の容量が少なくすみ、R A M の容量を小さくすることができる。

【0 0 8 1】

〔実施の形態 3〕

本実施の形態は、第 1 の実施の形態と異なり、圧縮するバイトコードの単位がメソッド単位ではなく命令単位である。

【0 0 8 2】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0 0 8 3】

また、ROM 4 に記憶されている情報は図 2 参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

#### 【 0 0 8 4 】

なお、圧縮バイトコード格納領域 1 2 には、命令単位でバイトコードが圧縮され、記憶されているものとする。

#### 【 0 0 8 5 】

図 9 を参照して、RAM 8 は、オペレーティングシステムを格納するオペレーティングシステム格納領域 1 6 と、バーチャルマシン（VM）モジュールを格納する VM モジュール格納領域 1 8 と、データの圧縮するためのプログラムである圧縮モジュールを格納するための圧縮モジュール格納領域 2 4 と、圧縮されたデータを伸張するためのプログラムである伸張モジュールを格納するための伸張モジュール格納領域 2 6 と、バイトコードおよびネイティブコード等を格納するためのコード格納領域 2 8 と、命令の状態を表わす命令ステータスを格納するための命令ステータス格納領域 9 0 とを含む。命令ステータス格納領域 9 0 は命令毎に設けられる。

#### 【 0 0 8 6 】

命令ステータス格納領域 9 0 は、命令が実行された回数を格納する回数格納領域 9 2 と、命令のサイズを格納するサイズ格納領域 9 4 と、ネイティブコードが存在するか否かを示すネイティブコード有無フラグを格納するネイティブコード有無フラグ格納領域 9 6 と、バイトコードがコンパイルされた時刻または順番を格納するコンパイル時刻（順番）格納領域 9 8 と、圧縮に関する情報を格納する圧縮情報格納領域 1 0 0 とを含む。

#### 【 0 0 8 7 】

圧縮情報格納領域 1 0 0 は、命令が圧縮されているか否かを示す圧縮フラグを格納する圧縮フラグ格納領域 1 0 2 と、圧縮された命令が伸張されているか否かを示す伸張済みフラグを格納する伸張済みフラグ格納領域 1 0 4 と、命令の圧縮方式を格納する圧縮方式格納領域 1 0 6 と、圧縮の際の圧縮率を格納する圧縮率格納領域 1 0 8 と、圧縮された時刻または順番を格納する圧縮時刻（順番）格納領域 1 1 0 と、圧縮ネイティブコードのサイズを格納する圧縮サイズ格納領域 1



1 2 とを含む。

【 0 0 8 8 】

図 1 0 を参照して、上位のモジュールよりある命令の呼出しがあった場合には、CPU 2 は、呼出された命令の伸張済みフラグ格納領域 1 0 4 に格納されている伸張済みフラグを参照し、呼出された命令のバイトコードが伸張されており、伸張済みバイトコード格納領域 3 0 に格納されているか否かを調べる ( S 7 2 )

【 0 0 8 9 】

バイトコードが伸張されていなければ ( S 7 2 で N O )、圧縮バイトコード格納領域 1 2 に格納されたバイトコードが伸張され ( S 7 4 )、伸張後のバイトコードが伸張済みバイトコード格納領域 3 0 に格納される ( S 7 6 )。次に、この命令のバイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される ( S 7 8 )。

【 0 0 9 0 】

S 7 8 の処理の後、または命令のバイトコードが伸張済みバイトコード格納領域 3 0 に格納されている場合には ( S 7 2 で Y E S )、CPU 2 はインタプリタを用いて、伸張済みのバイトコードを解釈し、実行する ( S 8 0 )。その後、呼出し側のモジュールへ復帰するための処理を行なう ( S 8 2 )。

【 0 0 9 1 】

なお、伸張済みバイトコード格納領域 3 0 に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応した命令の伸張済みフラグがオフに設定される。

【 0 0 9 2 】

以上説明したように、本実施の形態によると、バイトコードが圧縮されて R O M に記憶されている。このため、R O M の容量を小さくすることができる。

【 0 0 9 3 】

また、伸張済みバイトコード格納領域には、命令の圧縮バイトコードを伸張したバイトコードが格納される。このため、第 1 および第 2 の実施の形態に比べて伸張済みバイトコード格納領域の容量が少なくすみ、R A M の容量を小さくす

ることができる。

【0094】

〔実施の形態4〕

本実施の形態では、圧縮バイトコード格納領域に格納されている圧縮バイトコードの圧縮方法として、複数の圧縮方法がサポートされている。

【0095】

本実施の形態に係るJava（R）仮想マシンは、図1を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0096】

また、ROM4およびRAM8に記憶されている情報も図2および図3をそれぞれ参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0097】

なお、圧縮バイトコード格納領域12には、メソッド単位でバイトコードが圧縮され、記憶されているものとする。

【0098】

図11を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、CPU2は、呼出されたメソッドの伸張済みフラグ格納領域50に格納されている伸張済みフラグを参照し、呼出されたメソッドのバイトコードが、伸張されており、伸張済みバイトコード格納領域30に格納されているか否かを調べる（S92）。

【0099】

バイトコードが格納されていなければ（S92でNO）、圧縮バイトコード格納領域12に格納されたバイトコードの圧縮方式が判定される（S94）。圧縮方式の判定は圧縮バイトコードのヘッダ部分を参照することにより行なわれる。圧縮バイトコード格納領域12に格納されたバイトコードがS94で求められた圧縮方式に基づいて伸張され（S96）、伸張後のバイトコードが伸張済みバイトコード格納領域30格納される（S98）。次に、このメソッドのバイトコー

ドが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（S100）。

【0100】

S100の処理の後、またはメソッドのバイトコードが伸張済みバイトコード格納領域30に格納されている場合には（S92でYES）、CPU2はインタプリタを用いて、伸張済みのバイトコードを1命令ずつ解釈しながら実行する（S102）。その後、呼出し側のモジュールへの復帰をするための処理を行なう（S104）。

【0101】

なお、伸張済みバイトコード格納領域30に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応したメソッドの伸張済みフラグがオフに設定される。

【0102】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてROMに記憶されている。このため、ROMの容量を小さくすることができる。

【0103】

また、メソッド毎に圧縮方式をユーザが選択することができ、最適な圧縮方式を採用することができるため、ROMの容量を小さくすることができる。

【0104】

〔実施の形態5〕

本実施の形態ではJITを用いて、圧縮バイトコードを伸張後、ネイティブコードにコンパイルし、実行するものである。

【0105】

本実施の形態に係るJava（R）仮想マシンは、図1を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0106】

また、ROM4およびRAM8に記憶されている情報も図2および図3をそれぞれ参照して説明したものと同様である。このため、その詳細な説明はここでは

繰返さない。

【0107】

なお、圧縮バイトコード格納領域12には、メソッド単位でバイトコードが圧縮され、記憶されているものとする。

【0108】

図12を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、CPU2は、呼出されたメソッドの伸張済みフラグ格納領域50に格納されている伸張済みフラグを参照し、呼出されたメソッドのバイトコードが回答されており、かつネイティブコードに変換され、ネイティブコード格納領域32に格納されているか否かを調べる(S112)。

【0109】

バイトコードが伸張されていなければ(S112でNO)、圧縮バイトコード格納領域12に格納されたバイトコードが伸張され(S114)、伸張後のバイトコードがJIT格納領域22に格納されたJITによりコンパイルされ、ネイティブコードに変換される(S116)。ネイティブコードは、ネイティブコード格納領域32に格納される(S118)。このメソッドの圧縮バイトコードが伸張済みおよびコンパイル済みであることを表わすため、伸張済みフラグの値がオンに設定される(S120)。

【0110】

S120の処理の後、またはメソッドのネイティブコードがネイティブコード格納領域32に格納されている場合には(S112でYES)、CPU2は、プログラムカウンタの値をネイティブコードの先頭アドレスに設定し、ネイティブコードを実行する(S122)。その後、呼出し側のモジュールへ復帰するための処理を行なう(S124)。

【0111】

なお、ネイティブコード格納領域32に格納されていたネイティブコードが削除または他のネイティブコードの上書き等により失われる場合には、そのネイティブコードに対応したメソッドの伸張済みフラグがオフに設定される。

【0112】

図4を参照して、J a v a ( R ) 言語で記述されたプログラムが、メソッド1～4の4つのメソッドにより構成されるものとし、メソッド1、メソッド2、メソッド3、メソッド1、メソッド2、メソッド4、メソッド4、メソッド4の順に呼出されて、実行されるものとする。なお、ネイティブコード格納領域32には、4つのメソッドのネイティブコードを一度にすべて格納することができるものとする。このとき、図13を参照して、C P U 2は以下の順序で処理を実行する。

## 【0113】

メソッド1の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする(S132)。メソッド1のネイティブコードが1命令ずつ実行される(S134)。

## 【0114】

メソッド2の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする(S136)。メソッド2のネイティブコードが1命令ずつ実行される(S138)。

## 【0115】

メソッド3の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする(S140)。メソッド3のネイティブコードが1命令ずつ実行される(S142)。

## 【0116】

メソッド1のネイティブコードが1命令ずつ実行される(S144)。メソッド2のネイティブコードが1命令ずつ実行される(S146)。

## 【0117】

メソッド4の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする(S148)。メソッド4のネイティブコードが1命令ずつ実行される(S150)。メソッド4のネイティブコードが1命令ずつ実行される(S152)。メソッド4のネイティブコードが1命令ずつ実行される(S154)。

## 【0118】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてROMに記憶されている。このため、ROMの容量を小さくすることができる。

【0119】

なお、ネイティブコード格納領域32をCPU2とRAM8との間に設けられたキャッシュメモリ（図示せず）に設けることも可能である。このようにすれば、処理の実行速度を向上させることができる。

【0120】

〔実施の形態6〕

本実施の形態では、JITを用いてバイトコードを、ネイティブコードにコンパイルし、実行するものである。その際、不要になったネイティブコードを圧縮して保存しておく。

【0121】

本実施の形態に係るJava（R）仮想マシンは、図1を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0122】

また、RAM8に記憶されている情報は図3を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0123】

図14を参照して、ROM4は、Java（R）言語で記述されたプログラムのバイトコードを格納するバイトコード格納領域122と、その他のクラス情報を格納する領域124とを含む。

【0124】

図15を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、CPU2は、ネイティブコード有無フラグ格納領域42に格納されているそのメソッドのネイティブコード有無フラグを参照し、ネイティブコード格納領域32に呼出されたネイティブコードが格納されているか否かを判断する（S162）。

【0125】

ネイティブコード格納領域 3 2 にネイティブコードが格納されていなければ (S 1 6 2 で N O)、ネイティブコード格納領域 3 2 にネイティブコードを追加格納するだけの空きがあるか否か判断する (S 1 6 4)。

## 【 0 1 2 6 】

ネイティブコード格納領域 3 2 に空きがなければ (S 1 6 4 で N O)、現在、ネイティブコード格納領域 3 2 に存在するメソッドのネイティブコード有無フラグをオフにし、ネイティブコード格納領域 3 2 に空きを作る (S 1 6 6)。

## 【 0 1 2 7 】

ネイティブコード格納領域 3 2 に空きがあるか (S 1 6 4 で Y E S)、ネイティブコード格納領域 3 2 に空きを作った場合には (S 1 6 6)、圧縮ネイティブコード格納領域 3 4 に呼出されたメソッドの圧縮ネイティブコードが格納されているか否かを調べる (S 1 6 8)。すなわち、そのメソッドの圧縮フラグ格納領域 4 8 に格納された圧縮フラグがオンになっているか否かを調べる。

## 【 0 1 2 8 】

圧縮ネイティブコード格納領域 3 4 に呼出されたメソッドの圧縮ネイティブコードが格納されていれば (S 1 6 8 で Y E S)、圧縮ネイティブコードを伸張した後 (S 1 7 0)、ネイティブコード格納領域 3 2 に格納する (S 1 7 4)。

## 【 0 1 2 9 】

圧縮ネイティブコード格納領域 3 4 に呼出されたメソッドの圧縮ネイティブコードが格納されていなければ、そのメソッドのバイトコードを R O M 4 のバイトコード格納領域 1 2 2 より読出し、ネイティブコードに変換した後 (S 1 7 2)、ネイティブコード格納領域 3 2 に格納する (S 1 7 4)。

## 【 0 1 3 0 】

S 1 7 4 の処理の後、呼出されたメソッドのネイティブコード有無フラグをオンする。S 1 7 6 の処理の後、またはネイティブコード格納領域 3 2 に呼出されたメソッドのネイティブコードが格納されている場合には (S 1 6 2 で Y E S)、C P U 2 は、ネイティブコードを実行する (S 1 7 8)。

## 【 0 1 3 1 】

呼出されたメソッドの圧縮ネイティブコードが圧縮ネイティブコード格納領域

34に格納されているか否か判断される(S180)。圧縮ネイティブコードが圧縮ネイティブコード格納領域34に格納されていなければ、ネイティブコード格納領域32に格納されている、呼出されたメソッドのネイティブコードを圧縮し、圧縮ネイティブコード格納領域34に格納する(S182)。また、圧縮フラグ格納領域48に格納されているこのメソッドの圧縮フラグをオンにする(S184)。

#### 【0132】

圧縮ネイティブコードが圧縮ネイティブコード格納領域34に格納されているか(S180でYES)、圧縮ネイティブコード格納領域34に格納した場合には(S184)、呼出し側のモジュールへ復帰するための処理を行なう(S186)。

#### 【0133】

図4を参照して、Java(R)言語で記述されたプログラムが、メソッド1～4の4つのメソッドにより構成されるものとし、メソッド1、メソッド2、メソッド3、メソッド1、メソッド2、メソッド4、メソッド4、メソッド4の順に呼出されて、実行されるものとする。

#### 【0134】

なお、ネイティブコード格納領域32には、1つのメソッドのネイティブコードしか格納することができないものとし、圧縮ネイティブコード格納領域34にはすべてのメソッドの圧縮ネイティブコードを格納できるものとする。

#### 【0135】

このとき、図16および図17を参照して、CPU2は以下の順序で処理を実行する。

#### 【0136】

メソッド1をネイティブコード1に変換し、ネイティブコード格納領域32に格納する(S192)。ネイティブコード1を実行する(S194)。ネイティブコード1を圧縮し、圧縮ネイティブコード格納領域34に格納する(S196)。

#### 【0137】



メソッド2をネイティブコード2に変換し、ネイティブコード格納領域32に格納する(S198)。ネイティブコード2を実行する(S200)。ネイティブコード2を圧縮し、圧縮ネイティブコード格納領域34に格納する(S202)。

## 【0138】

メソッド3をネイティブコード3に変換し、ネイティブコード格納領域32に格納する(S204)。ネイティブコード3を実行する(S206)。ネイティブコード3を圧縮し、圧縮ネイティブコード格納領域34に格納する(S208)。

## 【0139】

圧縮ネイティブコード格納領域34に格納されているメソッド1の圧縮ネイティブコードを伸張し、実行する(S210)。圧縮ネイティブコード格納領域34に格納されているメソッド2の圧縮ネイティブコードを伸張し、実行する(S212)。

## 【0140】

メソッド4をネイティブコード4に変換し、ネイティブコード格納領域32に格納する(S214)。ネイティブコード4を実行する(S216)。ネイティブコード4を圧縮し、圧縮ネイティブコード格納領域34に格納する(S218)。ネイティブコード格納領域32に格納されているネイティブコード4を2回連続して実行する(S220)。

## 【0141】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、JITを用いた従来のJava(R)VMに比べ、RAMの使用容量を削減することが可能になる。

## 【0142】

## 〔実施の形態7〕

本実施の形態では、第6の実施の形態と異なり、ネイティブコードを圧縮するための圧縮方式として複数種類の圧縮方式が用意されているものとする。

## 【0143】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【 0 1 4 4 】

また、R A M 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 1 4 5 】

さらに、R O M 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 1 4 6 】

図 1 8 を参照して、上位モジュールよりあるメソッドの呼出しがあった場合の C P U 2 の行う処理について説明する。C P U 2 の行なう処理は、図 1 5 を参照して説明した処理において S 1 7 0 の処理の代わりに S 2 2 2 の処理を実行し、S 1 8 2 および S 1 8 4 の処理の代わりに、S 2 2 4 ~ S 2 2 8 の処理を実行するものである。

【 0 1 4 7 】

S 2 2 2 では、呼出されたメソッドの伸張を行なうが、どのような方式で圧縮されているかを圧縮方式格納領域 5 2 に格納された値から調べ、その方式に基づいたメソッドのネイティブコードの伸張を行なう。

【 0 1 4 8 】

S 2 2 4 では、所定の規則に基づいて、呼出されたメソッドのネイティブコードの圧縮に最適な方式を判定する。S 2 2 6 では、選択された圧縮方式でネイティブコードを圧縮し、圧縮ネイティブコード格納領域 3 4 に格納する。S 2 2 8 では、圧縮フラグ格納領域 4 8 に格納されているこのメソッドの圧縮フラグをオンにするとともに、圧縮方式を圧縮方式格納領域 5 2 に格納する。

【 0 1 4 9 】

図 4 を参照して、J a v a ( R ) 言語で記述されたプログラムが、メソッド 1 ~ 4 の 4 つのメソッドにより構成されるものとし、メソッド 1、メソッド 2、メソッド 3、メソッド 1、メソッド 2、メソッド 4、メソッド 4、メソッド 4 の順

に呼出されて、実行されるものとする。

【0150】

なお、ネイティブコード格納領域32には、1つのメソッドのネイティブコードしか格納することができないものとし、圧縮ネイティブコード格納領域34にはすべてのメソッドの圧縮ネイティブコードを格納できるものとする。

【0151】

また、圧縮方式には圧縮方式AおよびBの2種類があるものとし、メソッド1のネイティブコードの圧縮には圧縮方式Aが適しており、メソッド2～4のネイティブコードの圧縮には圧縮方式Bが適しているものとする。

【0152】

このとき、図19、図20および図21を参照して、CPU2は以下の順序で処理を実行する。

【0153】

メソッド1をネイティブコード1に変換し、ネイティブコード格納領域32に格納する(S232)。ネイティブコード1を実行する(S234)。ネイティブコード1を圧縮するのに最適な圧縮方式Aを選択し、ネイティブコード1を圧縮した後、圧縮ネイティブコード格納領域34に格納する(S236)。ネイティブコード1が圧縮方式Aで圧縮されたことを圧縮方式格納領域52に格納する(S238)。

【0154】

メソッド2をネイティブコード2に変換し、ネイティブコード格納領域32に格納する(S240)。ネイティブコード2を実行する(S242)。ネイティブコード2を圧縮するのに最適な圧縮方式Bを選択し、ネイティブコード2を圧縮した後、圧縮ネイティブコード格納領域34に格納する(S244)。ネイティブコード2が圧縮方式Bで圧縮されたことを圧縮方式格納領域52に格納する(S246)。

【0155】

メソッド3をネイティブコード3に変換し、ネイティブコード格納領域32に格納する(S248)。ネイティブコード3を実行する(S250)。ネイティ

ブコード 3 を圧縮するのに最適な圧縮方式 B を選択し、ネイティブコード 3 を圧縮した後、圧縮ネイティブコード格納領域 3 4 に格納する (S 2 5 2)。ネイティブコード 3 が圧縮方式 B で圧縮されたことを圧縮方式格納領域 5 2 に格納する (S 2 5 4)。

【0 1 5 6】

圧縮ネイティブコード格納領域 3 4 に格納されているメソッド 1 の圧縮ネイティブコードの圧縮方式を圧縮方式格納領域 5 2 に記憶されている値より調べる (S 2 5 6)。その結果、圧縮方式は、圧縮方式 A であることがわかる。圧縮ネイティブコード格納領域 3 4 に格納されているメソッド 1 の圧縮ネイティブコードを圧縮方式 A に対応した方式で伸張し、実行する (S 2 5 8)。

【0 1 5 7】

圧縮ネイティブコード格納領域 3 4 に格納されているメソッド 2 の圧縮ネイティブコードの圧縮方式を圧縮方式格納領域 5 2 に記憶されている値より調べる (S 2 6 0)。その結果、圧縮方式は、圧縮方式 B であることがわかる。圧縮ネイティブコード格納領域 3 4 に格納されているメソッド 2 の圧縮ネイティブコードを圧縮方式 B に対応した方式で伸張し、実行する (S 2 6 2)。

【0 1 5 8】

メソッド 4 をネイティブコード 4 に変換し、ネイティブコード格納領域 3 2 に格納する (S 2 6 4)。ネイティブコード 4 を実行する (S 2 6 6)。ネイティブコード 4 を圧縮するのに最適な圧縮方式 B を選択し、ネイティブコード 4 を圧縮した後、圧縮ネイティブコード格納領域 3 4 に格納する (S 2 6 8)。ネイティブコード 4 が圧縮方式 B で圧縮されたことを圧縮方式格納領域 5 2 に格納する (S 2 7 0)。ネイティブコード格納領域 3 2 に格納されているネイティブコード 4 を 2 回連続して実行する (S 2 7 2)。

【0 1 5 9】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、J I T を用いた従来の J a v a (R) V M に比べ、R A M の使用容量を削減することが可能になる。

【0 1 6 0】

また、メソッド毎に最適な圧縮方式を用いてネイティブコードの圧縮が行なわれる。このため、圧縮ネイティブコード格納領域 3 4 の容量を第 6 の実施の形態よりも小さくすることができる。

【0 1 6 1】

〔実施の形態 8〕

本実施の形態では、第 6 および第 7 の実施の形態と異なり、ネイティブコード格納領域 3 2 および圧縮ネイティブコード格納領域 3 4 の格納できるメソッドの数に制限が設けられているものとする。

【0 1 6 2】

本実施の形態に係る J a v a (R) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0 1 6 3】

また、R A M 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0 1 6 4】

さらに、R O M 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0 1 6 5】

図 2 2 および図 2 3 を参照して、上位モジュールよりあるメソッド N の呼出しがあった場合の C P U 2 の行う処理について説明する。C P U 2 は、ネイティブコード有無フラグ格納領域 4 2 に格納されているメソッド N のネイティブコード有無フラグを参照し、ネイティブコード格納領域 3 2 にメソッド N のネイティブコードが格納されているか否かを判断する (S 2 8 2)。

【0 1 6 6】

ネイティブコード格納領域 3 2 にメソッド N のネイティブコードが格納されていなければ (S 2 8 2 で N O)、ネイティブコード格納領域 3 2 にネイティブコードを追加格納するだけの空きがあるか否かを判断する (S 2 8 4)。

【0 1 6 7】

ネイティブコード格納領域 3 2 に空きがなければ (S 2 8 4 で N O)、コンパイル時刻 (順番) 格納領域 4 4 の値を調べ、ネイティブコード格納領域 3 2 内で最も古いメソッドを調べ、そのメソッドをメソッド A とする (S 2 8 6)。

## 【 0 1 6 8 】

メソッド A の圧縮ネイティブコードが圧縮ネイティブコード格納領域 3 4 に格納されているか否かを判断する (S 2 8 8)。メソッド A の圧縮ネイティブコードが圧縮ネイティブコード格納領域 3 4 に格納されていなければ (S 2 8 8 で N O)、圧縮ネイティブコード格納領域 3 4 の空きが十分か否かを調べる (S 2 9 0)。圧縮ネイティブコード格納領域 3 4 の空きが十分でなければ (S 2 9 0 で N O)、回数格納領域 3 8 に格納された回数に基づいて、圧縮ネイティブコード格納領域 3 4 内で最も実行頻度の低いメソッドを調べ、そのメソッドをメソッド B とする (S 2 9 2)。メソッド B の圧縮ネイティブコードを解放、すなわち削除し (S 2 9 4)、圧縮フラグ格納領域 4 8 に格納されたメソッド B の圧縮フラグをオフにする (S 2 9 6)。その後、S 2 9 0 に戻る。

## 【 0 1 6 9 】

圧縮ネイティブコード格納領域 3 4 の空きが十分であれば (S 2 9 0 で Y E S)、メソッド A のネイティブコードを圧縮し、圧縮ネイティブコード格納領域 3 4 に格納する (S 2 9 8)。また、圧縮フラグ格納領域 4 8 に格納されているメソッド A の圧縮フラグをオンにする (S 3 0 0)。

## 【 0 1 7 0 】

メソッド A の圧縮ネイティブコードが圧縮ネイティブコード格納領域 3 4 に格納されている場合 (S 2 8 8 で Y E S)、または S 3 0 0 の処理の後、ネイティブコード格納領域 3 2 のうち、メソッド A のネイティブコードが格納されている領域を解放し (S 3 0 2)、ネイティブコード有無フラグ格納領域 4 2 に格納されたメソッド A のネイティブコード有無フラグをオフにする (S 3 0 4)。その後、S 2 8 4 に戻る。

## 【 0 1 7 1 】

ネイティブコード格納領域 3 2 にメソッド N のネイティブコードを追加格納するだけの空きがあれば (S 2 8 4 で Y E S)、メソッド N のバイトコードをネイ

タイプコードに変換した後（S 3 0 6）、ネイティブコード格納領域 3 2 に格納する（S 3 0 8）。その後、メソッド N のネイティブコード有無フラグをオンにし、メソッド N がコンパイルされた順番または時刻をコンパイル時刻（順番）格納領域 4 4 に記憶する（S 3 1 0）。

【 0 1 7 2 】

ネイティブコード格納領域 3 2 にメソッド N のネイティブコードが格納されている場合（S 2 8 2 で Y E S）、または S 3 1 0 の処理の後、CPU 2 は、メソッド N のネイティブコードを実行する（S 3 1 2）。また、回数格納領域 3 8 に格納されたメソッド N の実行回数を 1 つインクリメントする（S 3 1 4）。その後、メソッド N の呼出し側へ戻るための処理を行なう（S 3 1 6）。

【 0 1 7 3 】

図 2 4 を参照して、J a v a ( R ) 言語で記述されたプログラムがメソッド 1 ～ 7 の 7 つのメソッドにより構成されるものとし、メソッド 1、メソッド 2、メソッド 3、メソッド 1、メソッド 2、メソッド 4、メソッド 5、メソッド 6、メソッド 7 の順に呼出されて、実行されるものとする。

【 0 1 7 4 】

なお、ネイティブコード格納領域 3 2 に格納できるメソッドの数は 3 つであり、圧縮ネイティブコード格納領域 3 4 に格納できるメソッドの数は 3 つであるものとする。

【 0 1 7 5 】

このとき、図 2 5、図 2 6 および図 2 7 を参照して、CPU 2 は以下の順序で処理を実行する。

【 0 1 7 6 】

メソッド 1 をネイティブコード 1 に変換し、ネイティブコード格納領域 3 2 に格納する（S 3 2 2）。ネイティブコード 1 を実行する（S 3 2 4）。メソッド 2 をネイティブコード 2 に変換し、ネイティブコード格納領域 3 2 に格納する（S 3 2 6）。ネイティブコード 2 を実行する（S 3 2 8）。メソッド 3 をネイティブコード 3 に変換し、ネイティブコード格納領域 3 2 に格納する（S 3 3 0）。メソッド 3 を実行する（S 3 3 2）。この時点で、ネイティブコード格納領域

32には3つのメソッドのネイティブコードが記憶されている。このため、ネイティブコード1～3以外のネイティブコードを格納するためには、いずれかのネイティブコードが確保している領域を解放しなければならない。

【0177】

CPU2は、ネイティブコード1を実行し（S334）、ネイティブコード2を実行する（S336）。

【0178】

メソッド4を実行するに際し、CPU2は、ネイティブコード格納領域32に格納されているネイティブコードの中で最も古いネイティブコード1を圧縮し、圧縮ネイティブコード格納領域34に格納する。それとともに、ネイティブコード格納領域32のネイティブコード1が格納されていた領域を解放する（S338）。メソッド4をネイティブコードに変換し、ネイティブコード格納領域32のネイティブコード1が格納されていた領域に格納する（S340）。ネイティブコード4を実行する（S342）。

【0179】

メソッド5を実行するに際し、CPU2は、ネイティブコード格納領域32に格納されているネイティブコードの中で最も古いネイティブコード2を圧縮し、圧縮ネイティブコード格納領域34に格納する。それとともに、ネイティブコード格納領域32のネイティブコード2が格納されていた領域を解放する（S344）。メソッド5をネイティブコードに変換し、ネイティブコード格納領域32のネイティブコード2が格納されていた領域に格納する（S346）。ネイティブコード5を実行する（S348）。

【0180】

メソッド6を実行するに際し、CPU2は、ネイティブコード格納領域32に格納されているネイティブコードの中で最も古いネイティブコード3を圧縮し、圧縮ネイティブコード格納領域34に格納する。それとともに、ネイティブコード格納領域32のネイティブコード3が格納されていた領域を解放する（S350）。メソッド6をネイティブコードに変換し、ネイティブコード格納領域32のネイティブコード3が格納されていた領域に格納する（S346）。ネイティ



ブコード6を実行する（S348）。

【0181】

図27を参照して、ここまでの時点でネイティブコード格納領域32には、ネイティブコード4～7が格納されており、圧縮ネイティブコード格納領域34には、ネイティブコード1～3が格納されている。

【0182】

図26を参照して、CPU2は、メソッド7のバイトコードをネイティブコードに変換する前にネイティブコード格納領域32に空き領域があるか否かを調べる（S356）。ネイティブコード格納領域32には空き領域がないため、ネイティブコード格納領域32に格納されているネイティブコードのうち、最も古いネイティブコード4を圧縮対象とする（S358）。

【0183】

CPU2は、ネイティブコード4を圧縮する前に、圧縮ネイティブコード格納領域34に空きがあるか否かを調べる（S360）。圧縮ネイティブコード格納領域34に空きがないため、圧縮ネイティブコード格納領域34に格納されている圧縮ネイティブコードのうち実行頻度の最も低い圧縮ネイティブコード3の領域を解放する（S362）。CPU2は、ネイティブコード4を圧縮し、圧縮ネイティブコード格納領域34の圧縮ネイティブコード3が格納されていた領域に格納する。それとともに、ネイティブコード格納領域32のネイティブコード4が格納されていた領域を解放する（S364）。

【0184】

CPU2は、メソッド7のバイトコードをネイティブコード7に変換し、ネイティブコード格納領域32のメソッド7が格納されていた領域に格納する（S366）。CPU2は、ネイティブコード7を実行する（S368）。

【0185】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、JITを用いた従来のJava（R）VMに比べ、RAMの使用容量を削減することが可能になる。

【0186】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

【0187】

〔実施の形態 9〕

本実施の形態は、第 8 の実施の形態と異なり、ネイティブコード格納領域 3 2 に格納しきれなくなったネイティブコードを圧縮する際のネイティブコードの選択基準として、最も実行頻度の低いネイティブコードを選択するものである。

【0188】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【0189】

また、RAM 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0190】

さらに、ROM 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0191】

図 2 8 および図 2 9 を参照して、上位モジュールよりメソッド N の呼出しがあった場合の C P U 2 の行なう処理について説明する。C P U 2 の行なう処理は、図 2 2 の S 2 8 6 の代わりに、図 2 8 の S 3 7 2 の処理を実行し、図 2 3 の S 3 1 0 の処理の代わりに、図 2 9 の S 3 7 4 の処理を実行するものである。その他の処理は、図 2 2 および図 2 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0192】

図 2 8 の S 3 7 2 では、回数格納領域 3 8 に格納されたネイティブコードの実行回数を調べ、ネイティブコード格納領域 3 2 内で最も実行頻度の低いメソッドを調べ、そのメソッド A とする。

【0193】

図 2 9 の S 3 7 4 では、メソッド N のネイティブコード有無フラグをオンにする。

【 0 1 9 4 】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、J I T を用いた従来の J a v a ( R ) V M に比べ、R A M の使用容量を削減することが可能になる。

【 0 1 9 5 】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

【 0 1 9 6 】

[ 実施の形態 1 0 ]

本実施の形態は、第 8 の実施の形態と異なり、ネイティブコード格納領域 3 2 に格納しきれなくなったネイティブコードを圧縮する際のネイティブコードの選択基準として、最もサイズの大きいネイティブコードを選択するものである。

【 0 1 9 7 】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【 0 1 9 8 】

また、R A M 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 1 9 9 】

さらに、R O M 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 0 0 】

図 3 0 および図 3 1 を参照して、上位モジュールよりメソッド N の呼出しがあった場合の C P U 2 の行なう処理について説明する。C P U 2 の行なう処理は、図 2 2 の S 2 8 6 の代わりに、図 3 0 の S 3 8 2 の処理を実行し、図 2 3 の S 3 1 0 の処理の代わりに、図 3 1 の S 3 8 4 の処理を実行するものである。その他

の処理は、図 2 2 および図 2 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

## 【0201】

図 3 0 の S 3 8 2 では、サイズ格納領域 4 0 に格納されたネイティブコードのサイズを調べることにより、ネイティブコード格納領域 3 2 内で最も大きいサイズのネイティブコードを有するメソッドを調べ、メソッド A とする。

## 【0202】

図 3 1 の S 3 8 4 では、メソッド N のネイティブコード有無フラグをオンにし、メソッド N のネイティブコードのサイズをサイズ格納領域 4 0 に格納する。

## 【0203】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、JIT を用いた従来の Java (R) VM に比べ、RAM の使用容量を削減することが可能になる。

## 【0204】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

## 【0205】

## 〔実施の形態 1 1〕

本実施の形態は、第 8 の実施の形態と異なり、ネイティブコード格納領域 3 2 に格納しきれなくなったネイティブコードを圧縮する際のネイティブコードの選択基準として、最も圧縮率の大きいネイティブコードを選択するものである。

## 【0206】

本実施の形態に係る Java (R) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

## 【0207】

また、RAM 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

## 【0208】

さらに、ROM 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

#### 【 0 2 0 9 】

図 3 2 および図 3 3 を参照して、上位モジュールよりメソッド N の呼出しがあった場合の CPU 2 の行なう処理について説明する。CPU 2 の行なう処理は、図 2 2 の S 2 8 6 の代わりに、図 3 2 の S 3 9 2 の処理を実行し、図 2 3 の S 3 1 0 の処理の代わりに、図 3 3 の S 3 9 4 の処理を実行するものである。その他の処理は、図 2 2 および図 2 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

#### 【 0 2 1 0 】

図 3 2 の S 3 9 2 では、圧縮率格納領域 5 4 に格納されたネイティブコードを圧縮した際の圧縮率を調べることにより、ネイティブコード格納領域 3 2 内で最も圧縮率の大きいネイティブコードを有するメソッドを調べ、メソッド A とする。

#### 【 0 2 1 1 】

図 3 3 の S 3 9 4 では、メソッド N のネイティブコード有無フラグをオンにし、メソッド N のネイティブコードを圧縮した際の圧縮率を圧縮率格納領域 5 4 に格納する。

#### 【 0 2 1 2 】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、JIT を用いた従来の Java (R) VM に比べ、RAM の使用容量を削減することが可能になる。

#### 【 0 2 1 3 】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

#### 【 0 2 1 4 】

##### 〔実施の形態 1 2〕

本実施の形態は、第 8 の実施の形態と異なり、圧縮ネイティブコード格納領域 3 4 に格納しきれなくなった圧縮ネイティブコードを解放する際の圧縮ネイティ

ブコードの選択基準として、最もサイズの大きい圧縮ネイティブコードを選択するものである。

【 0 2 1 5 】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【 0 2 1 6 】

また、R A M 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 1 7 】

さらに、R O M 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 1 8 】

図 3 4 および図 3 5 を参照して、上位モジュールよりメソッド N の呼出しがあった場合の C P U 2 の行なう処理について説明する。C P U 2 の行なう処理は、図 2 2 の S 2 9 2 の処理の代わりに、図 3 4 の S 4 0 2 の処理を実行し、図 2 3 の S 3 1 4 の処理の代わりに、図 3 5 の S 4 0 4 の処理を実行するものである。その他の処理は、図 2 2 および図 2 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 1 9 】

図 3 4 の S 4 0 2 では、圧縮サイズ格納領域 5 8 に格納された圧縮ネイティブコードのサイズを調べることにより、圧縮ネイティブコード格納領域 3 4 内で最もサイズの大きい圧縮ネイティブコードを有するメソッドを調べ、メソッド A とする。

【 0 2 2 0 】

図 3 5 の S 4 0 4 では、メソッド N のネイティブコードを圧縮した際のサイズを調べ、そのサイズを圧縮サイズ格納領域 5 8 に格納する。

【 0 2 2 1 】

以上説明したように、本実施の形態によると、バイトコードがネイティブコー

ドに変換された後、圧縮されて記憶される。このため、J I Tを用いた従来の J a v a ( R ) V M に比べ、R A M の使用容量を削減することが可能になる。

【 0 2 2 2 】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

【 0 2 2 3 】

〔実施の形態 1 3〕

本実施の形態は、第 8 の実施の形態と異なり、圧縮ネイティブコード格納領域 3 4 に格納しきれなくなった圧縮ネイティブコードを解放する際の圧縮ネイティブコードの選択基準として、圧縮率が最も低い圧縮ネイティブコードを選択するものである。

【 0 2 2 4 】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【 0 2 2 5 】

また、R A M 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 2 6 】

さらに、R O M 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 2 7 】

図 3 6 および図 3 7 を参照して、上位モジュールよりメソッド N の呼出しがあった場合の C P U 2 の行なう処理について説明する。C P U 2 の行なう処理は、図 2 2 の S 2 9 2 の処理の代わりに、図 3 6 の S 4 1 2 の処理を実行し、図 2 3 の S 3 1 4 の処理の代わりに、図 3 7 の S 4 1 4 の処理を実行するものである。その他の処理は、図 2 2 および図 2 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 2 8 】

図 3 6 の S 4 1 2 では、圧縮率格納領域 5 4 に格納されたネイティブコードの圧縮率を調べることにより、圧縮ネイティブコード格納領域 3 4 内で最も圧縮率の低い圧縮ネイティブコードを有するメソッドを調べ、メソッド A とする。

【 0 2 2 9 】

図 3 7 の S 4 1 4 では、メソッド N のネイティブコードを圧縮した際の圧縮率を調べ、圧縮率格納領域 5 4 に格納する。

【 0 2 3 0 】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、J I T を用いた従来の J a v a ( R ) V M に比べ、R A M の使用容量を削減することが可能になる。

【 0 2 3 1 】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

【 0 2 3 2 】

〔実施の形態 1 4 〕

本実施の形態は、第 8 の実施の形態と異なり、圧縮ネイティブコード格納領域 3 4 に格納しきれなくなった圧縮ネイティブコードを解放する際の圧縮ネイティブコードの選択基準として、最も古く圧縮された圧縮ネイティブコードを選択するものである。

【 0 2 3 3 】

本実施の形態に係る J a v a ( R ) 仮想マシンは、図 1 を参照して説明した組み込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【 0 2 3 4 】

また、R A M 8 に記憶されている情報は図 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【 0 2 3 5 】

さらに、R O M 4 に記憶されている情報は図 1 4 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。



## 【 0 2 3 6 】

図 3 8 および図 3 9 を参照して、上位モジュールよりメソッド N の呼出しがあった場合の CPU 2 の行なう処理について説明する。CPU 2 の行なう処理は、図 2 2 の S 2 9 2 および S 2 9 8 の処理の代わりに、図 3 8 の S 4 2 2 および S 4 2 4 の処理をそれぞれ行なうものである。また、図 2 3 の S 3 1 4 の処理を省略したもの（図 3 9 の処理）である。その他の処理は、図 2 2 および図 2 3 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

## 【 0 2 3 7 】

図 3 8 の S 4 2 2 では、圧縮時刻（順番）格納領域 5 6 に格納された圧縮時刻を調べることにより、圧縮ネイティブコード格納領域 3 4 内で最初に圧縮された圧縮ネイティブコードを有するメソッドを調べ、メソッド A とする。

## 【 0 2 3 8 】

図 3 8 の S 4 2 4 では、メソッド A のネイティブコードを圧縮し、圧縮ネイティブコード格納領域 3 4 に格納する。その際、圧縮した時刻を圧縮時刻（順番）格納領域 5 6 に格納する。

## 【 0 2 3 9 】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、JIT を用いた従来の Java (R) VM に比べ、RAM の使用容量を削減することが可能になる。

## 【 0 2 4 0 】

また、ネイティブコード格納領域 3 2 に複数のメソッドのネイティブコードを格納することができるため、第 7 の実施の形態に比べて、処理が高速である。

## 【 0 2 4 1 】

なお、実施の形態 1 ～ 1 4 に記載の発明は、Java (R) 言語以外のオブジェクト指向言語で記載され、かつ中間言語にコンパイルされたプログラムにも適用可能である。

## 【 0 2 4 2 】

今回開示された実施の形態はすべての点で例示であって制限的なものではない

と考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【 0 2 4 3 】

【発明の効果】

オブジェクト指向言語のプログラムを実行する際に必要なメモリの容量を小さくすることができる。

【図面の簡単な説明】

【図 1】 本発明の実施の形態に係る組込み機器のハードウェア構成を示すブロック図である。

【図 2】 ROMに格納されているデータを説明するための図である。

【図 3】 RAMに格納されているデータを説明するための図である。

【図 4】 J a v a ( R ) 言語で記述されたプログラムの構成を示す図である。

【図 5】 実施の形態 1 において、メソッドが呼出された場合の処理のフローチャートである。

【図 6】 実施の形態 1 において、メソッドが呼出された場合の処理のフローチャートである。

【図 7】 RAMに格納されているデータを説明するための図である。

【図 8】 実施の形態 2 において、基本ブロックが呼出された場合の処理のフローチャートである。

【図 9】 RAMに格納されるデータを説明するための図である。

【図 1 0】 実施の形態 3 において、命令が呼出された場合の処理のフローチャートである。

【図 1 1】 実施の形態 4 において、メソッドが呼出された場合の処理のフローチャートである。

【図 1 2】 実施の形態 5 において、メソッドが呼出された場合の処理のフローチャートである。

【図 1 3】 実施の形態 5 において、メソッドが呼出された場合の処理のフ

ローチャートである。

【図 1 4】 ROMに格納されているデータを説明するための図である。

【図 1 5】 実施の形態 6 において、メソッドが呼出された場合の処理のフローチャートである。

【図 1 6】 実施の形態 6 において、メソッドが呼出された場合の処理のフローチャートである。

【図 1 7】 実施の形態 6 において、メソッドが呼出された場合の処理を説明するための図である。

【図 1 8】 実施の形態 7 において、メソッドが呼出された場合の処理のフローチャートである。

【図 1 9】 実施の形態 7 において、メソッドが呼出された場合の処理のフローチャートである。

【図 2 0】 実施の形態 7 において、メソッドが呼出された場合の処理のフローチャートである。

【図 2 1】 実施の形態 7 において、メソッドが呼出された場合の処理を説明するための図である。

【図 2 2】 実施の形態 8 において、メソッドが呼出された場合の処理のフローチャートである。

【図 2 3】 実施の形態 8 において、メソッドが呼出された場合の処理のフローチャートである。

【図 2 4】 J a v a ( R ) 言語で記述されたプログラムの構成を示す図である。

【図 2 5】 実施の形態 8 において、メソッドが呼出された場合の処理のフローチャートである。

【図 2 6】 実施の形態 8 において、メソッドが呼出された場合の処理のフローチャートである。

【図 2 7】 実施の形態 8 において、メソッドが呼出された場合の処理を説明するための図である。

【図 2 8】 実施の形態 9 において、メソッドが呼出された場合の処理のフ

ローチャートである。

【図 2 9】 実施の形態 9 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 0】 実施の形態 1 0 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 1】 実施の形態 1 0 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 2】 実施の形態 1 1 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 3】 実施の形態 1 1 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 4】 実施の形態 1 2 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 5】 実施の形態 1 2 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 6】 実施の形態 1 3 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 7】 実施の形態 1 3 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 8】 実施の形態 1 4 において、メソッドが呼出された場合の処理のフローチャートである。

【図 3 9】 実施の形態 1 4 において、メソッドが呼出された場合の処理のフローチャートである。

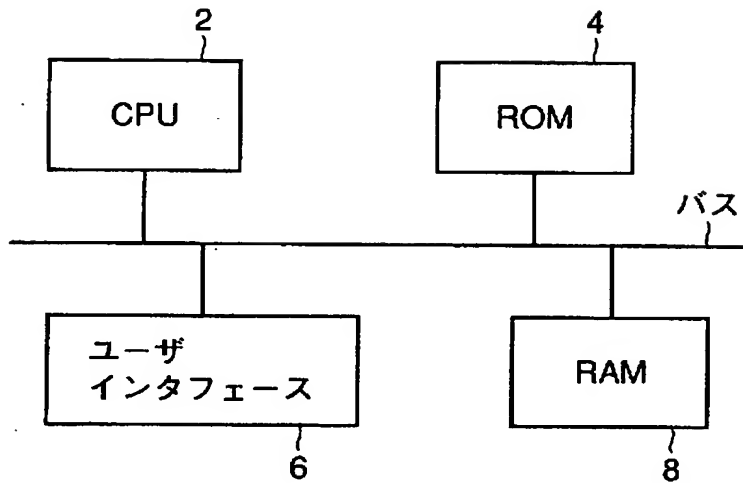
#### 【符号の説明】

2 CPU、4 ROM、6 ユーザインタフェース、12 圧縮バイトコード格納領域、14、124 領域、16 オペレーティングシステム格納領域、18 モジュール格納領域、20 インタプリタ格納領域、22 JIT格納領域、24 圧縮モジュール格納領域、26 伸張モジュール格納領域、28 コード格納領域、30 バイトコード格納領域、32 ネイティブコード格納領域

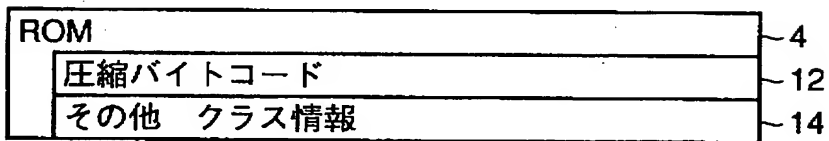
、34 圧縮ネイティブコード格納領域、36 メソッドステータス格納領域、  
38, 62, 92 回数格納領域、40, 64, 94 サイズ格納領域、42,  
66, 96 ネイティブコード有無フラグ格納領域、44, 68, 98 コンパ  
イル時刻(順番)格納領域、46, 70, 100 圧縮情報格納領域、48, 7  
2, 102 圧縮フラグ格納領域、50, 74, 104 伸張済みフラグ格納領  
域、52, 76, 106 圧縮方式格納領域、54, 78, 108 圧縮率格納  
領域、56, 80, 110 圧縮時刻(順番)格納領域、58, 82, 112  
圧縮サイズ格納領域、60 基本ブロックステータス格納領域、90 命令ステ  
ータス格納領域、122 バイトコード格納領域。

【書類名】 図面

【図 1】



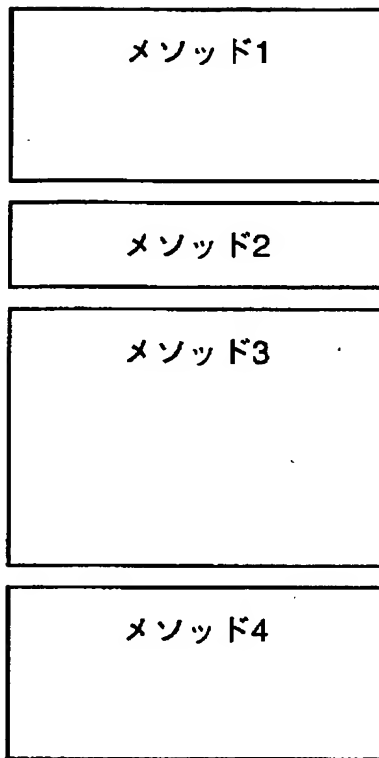
【図 2】



【図 3】

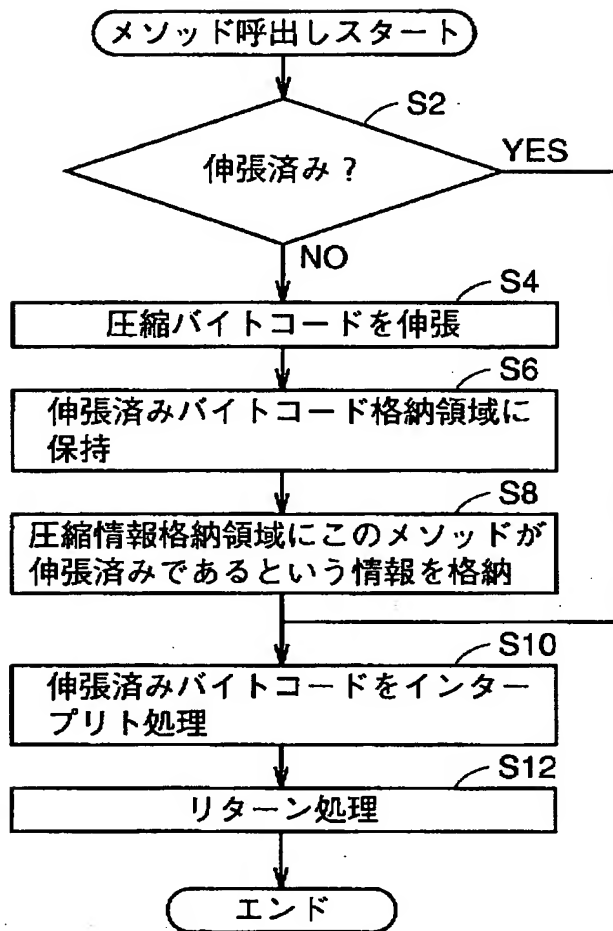
RAM	8
オペレーティングシステム	16
バーチャルマシン (VM) モジュール	18
インタプリタ	20
JIT	22
圧縮モジュール	24
伸張モジュール	26
コード	28
伸張済みバイトコード	30
ネイティブコード	32
圧縮ネイティブコード	34
メソッドステータス	36
実行された回数	38
サイズ	40
ネイティブコード有無フラグ	42
コンパイルされた時刻 (順番)	44
圧縮情報	46
圧縮フラグ	48
伸張済みフラグ	50
圧縮方式	52
圧縮率	54
圧縮された時刻 (順番)	56
圧縮サイズ	58

【図 4】

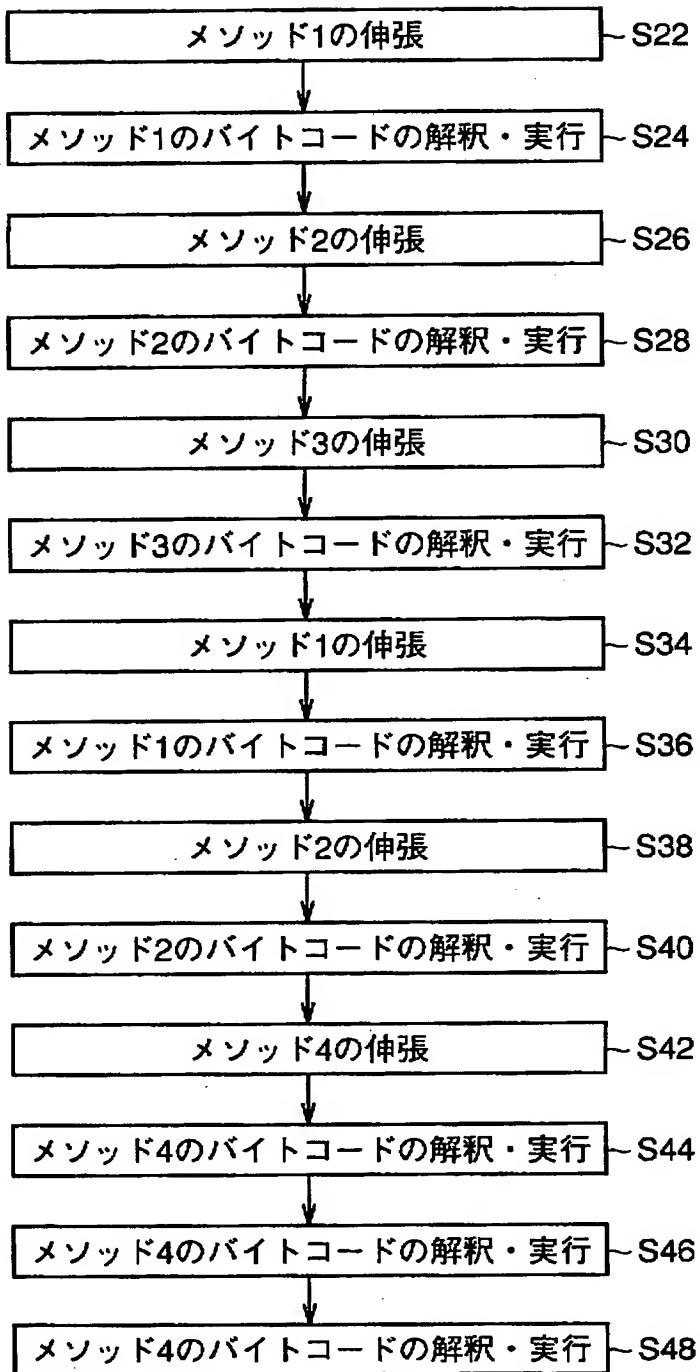




【図 5】



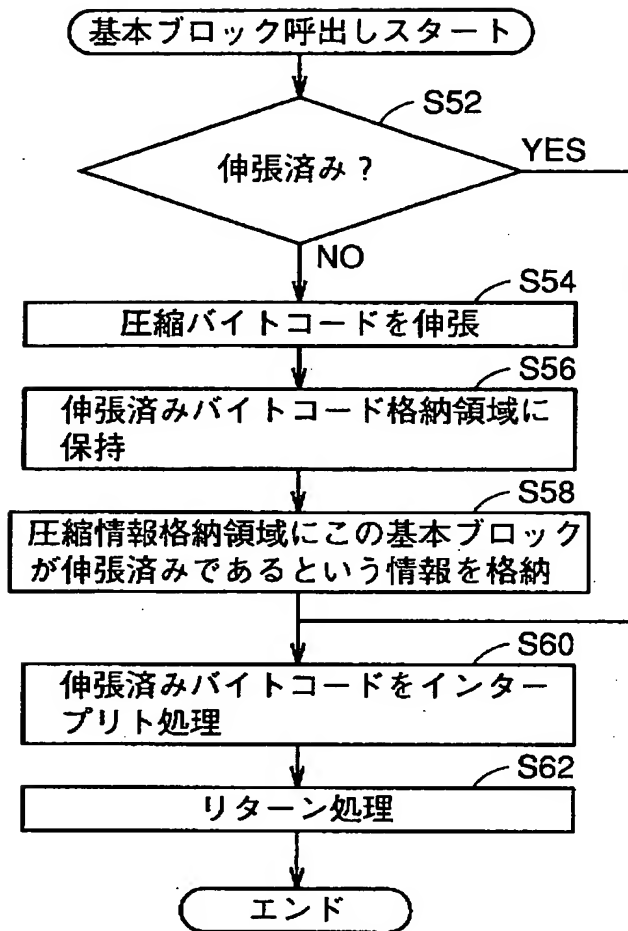
【図 6】



【図 7】

RAM	~8
オペレーティングシステム	~16
バーチャルマシン (VM) モジュール	~18
インタプリタ	~20
JIT	~22
圧縮モジュール	~24
伸張モジュール	~26
コード	~28
伸張済みバイトコード	~30
ネイティブコード	~32
圧縮ネイティブコード	~34
基本ブロックステータス	~60
実行された回数	~62
サイズ	~64
ネイティブコード有無フラグ	~66
コンパイルされた時刻 (順番)	~68
圧縮情報	~70
圧縮フラグ	~72
伸張済みフラグ	~74
圧縮方式	~76
圧縮率	~78
圧縮された時刻 (順番)	~80
圧縮サイズ	~82

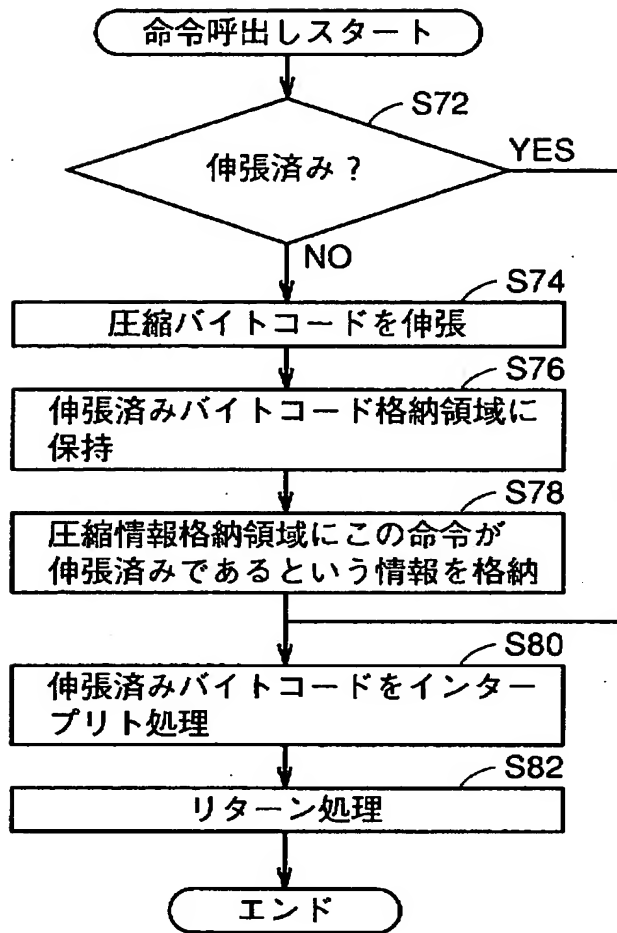
【図 8】



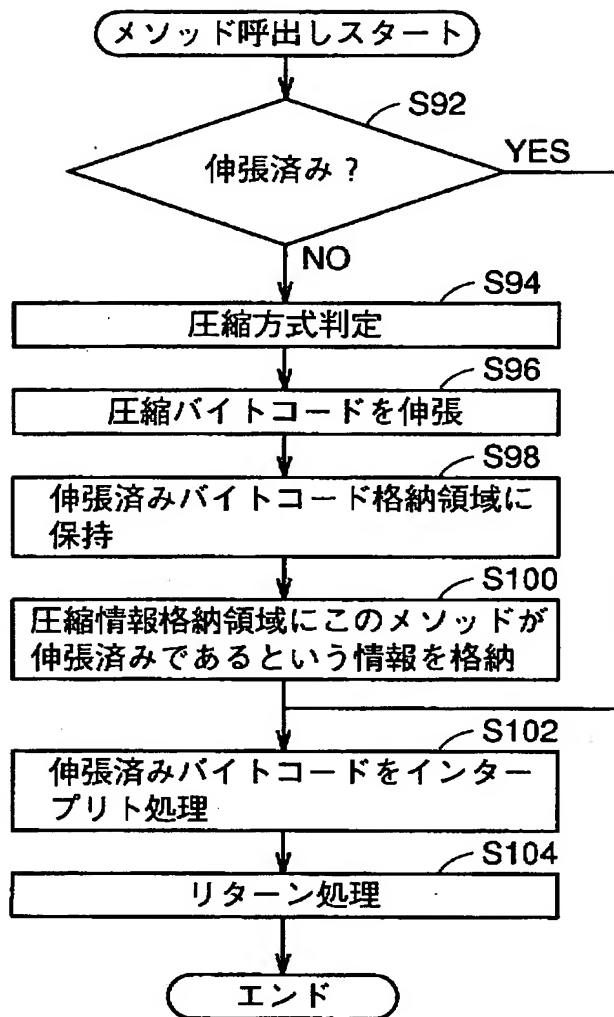
【図 9】

RAM	8
オペレーティングシステム	16
バーチャルマシン (VM) モジュール	18
インタプリタ	20
JIT	22
圧縮モジュール	24
伸張モジュール	26
コード	28
伸張済みバイトコード	30
ネイティブコード	32
圧縮ネイティブコード	34
命令ステータス	90
実行された回数	92
サイズ	94
ネイティブコード有無フラグ	96
コンパイルされた時刻 (順番)	98
圧縮情報	100
圧縮フラグ	102
伸張済みフラグ	104
圧縮方式	106
圧縮率	108
圧縮された時刻 (順番)	110
圧縮サイズ	112

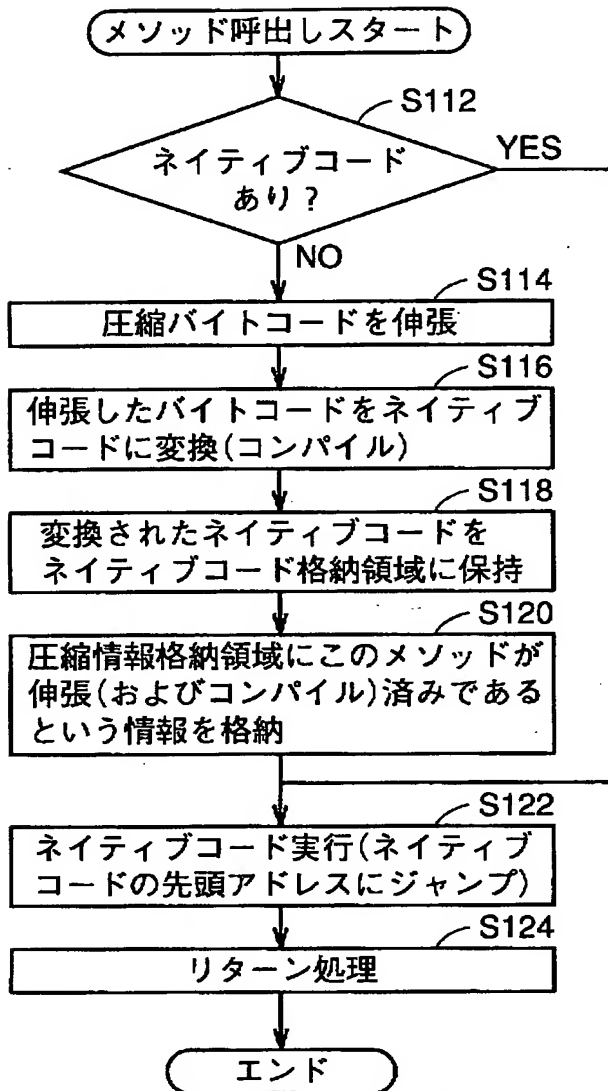
【図 1 0】



【図 1 1】

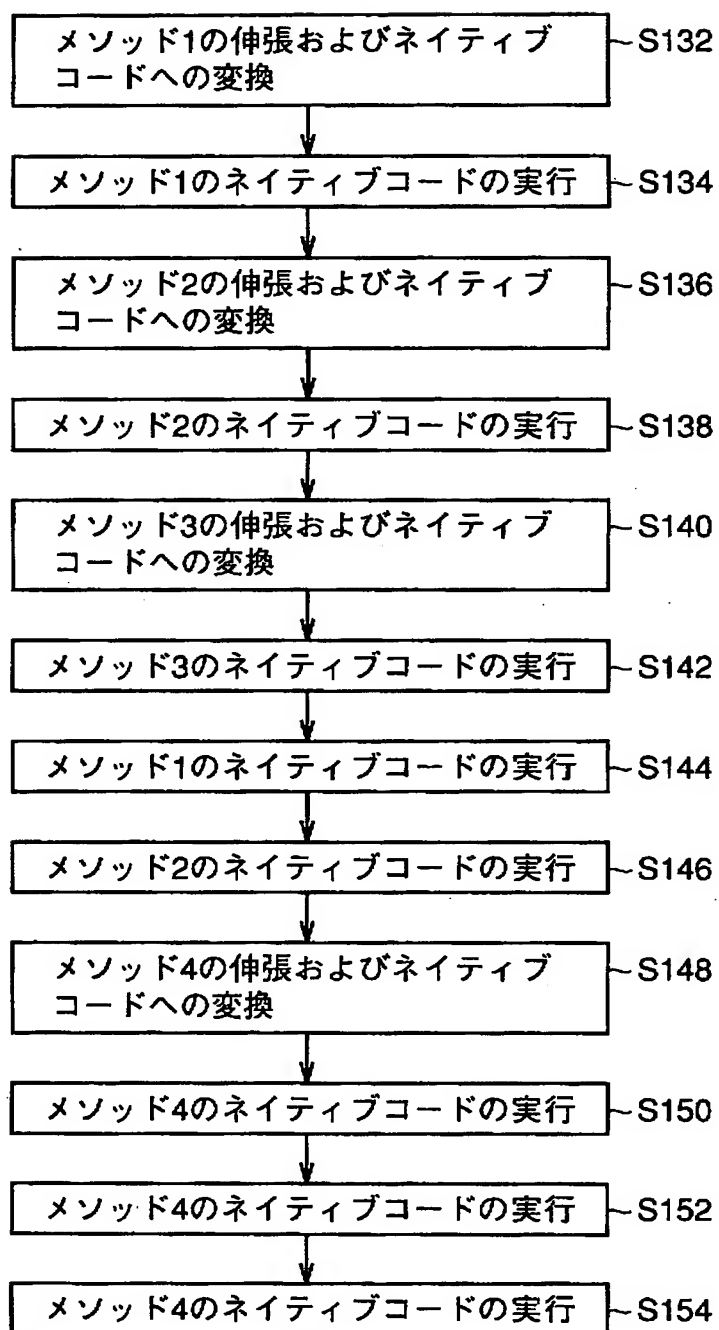


【図 12】





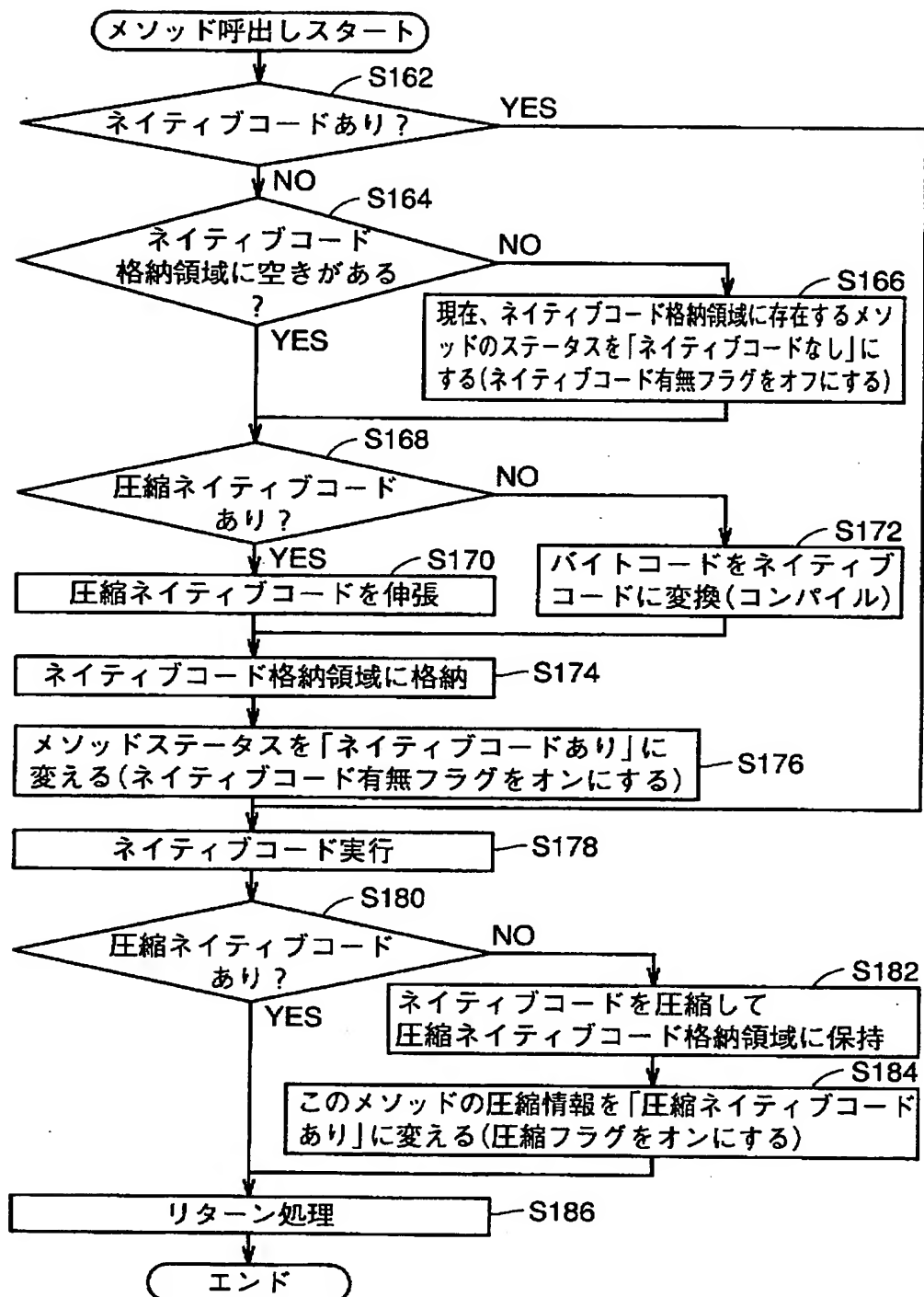
【図 1 3】



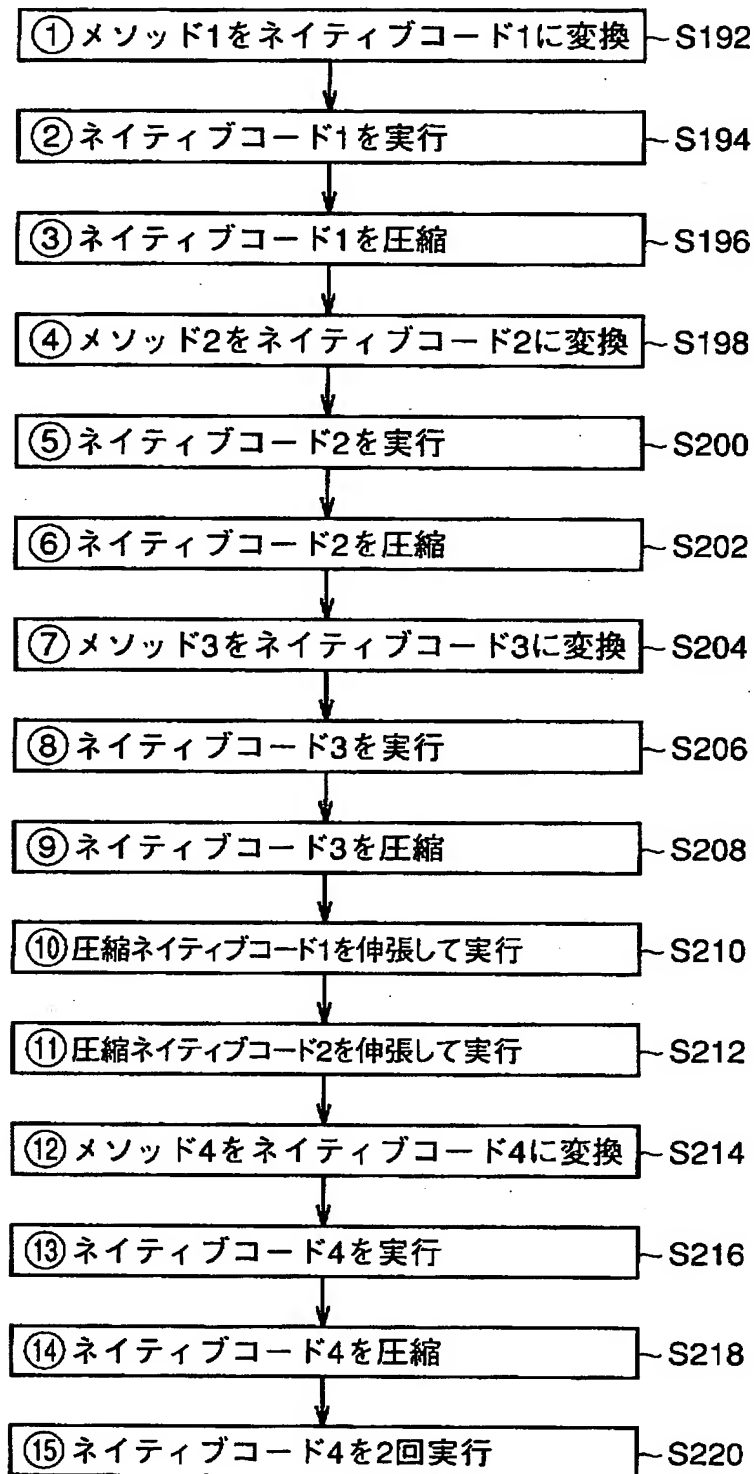
【図 1 4】

ROM	4
バイトコード	122
その他 クラス情報	124

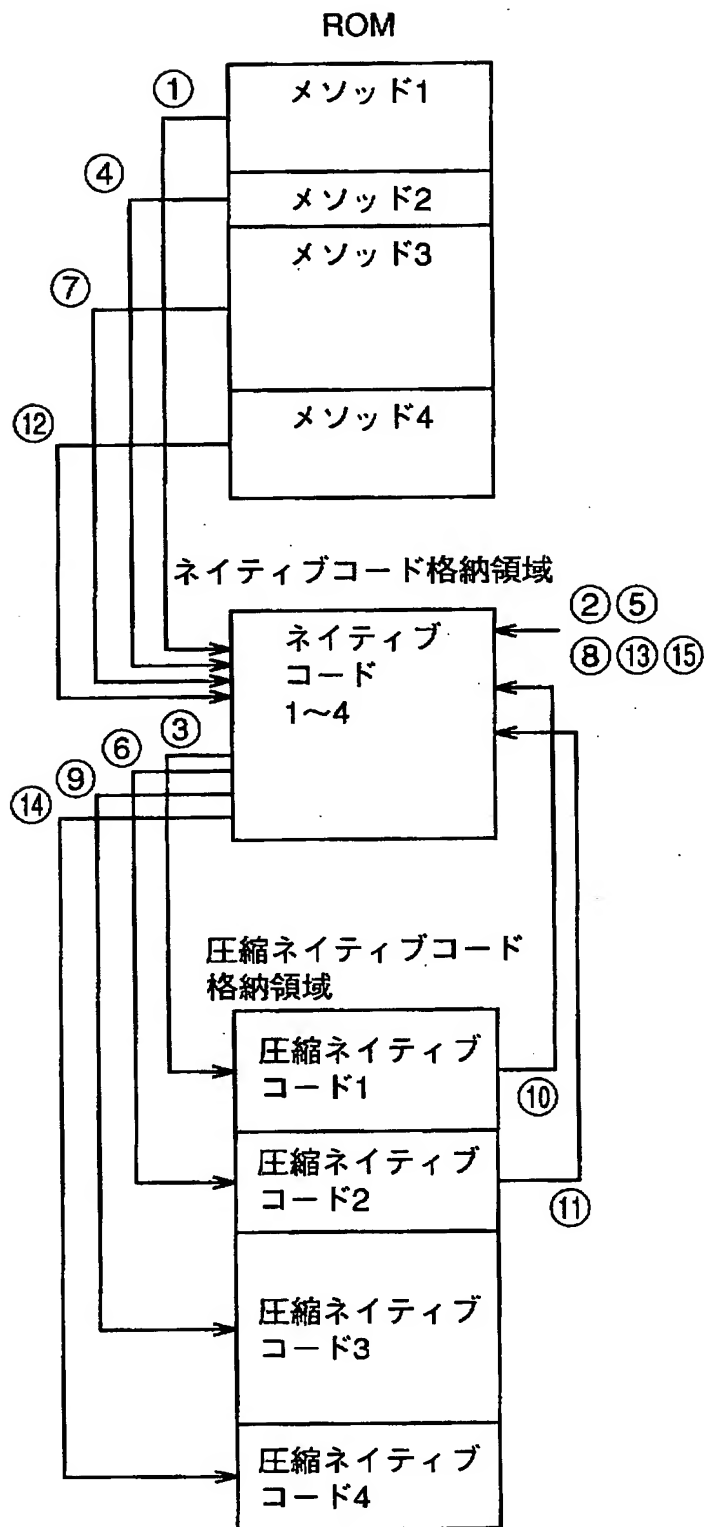
【図 1 5】



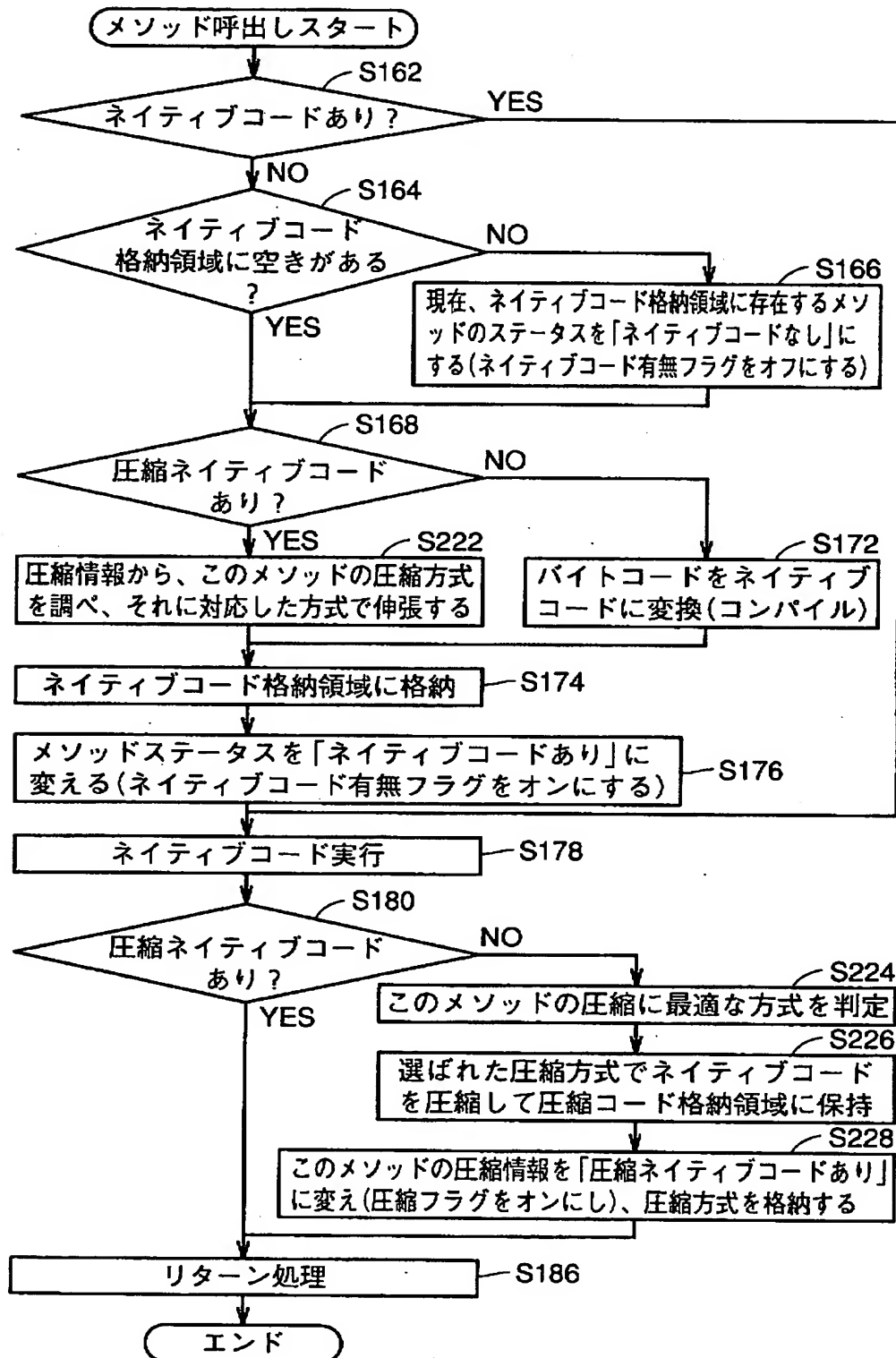
【図 1 6】



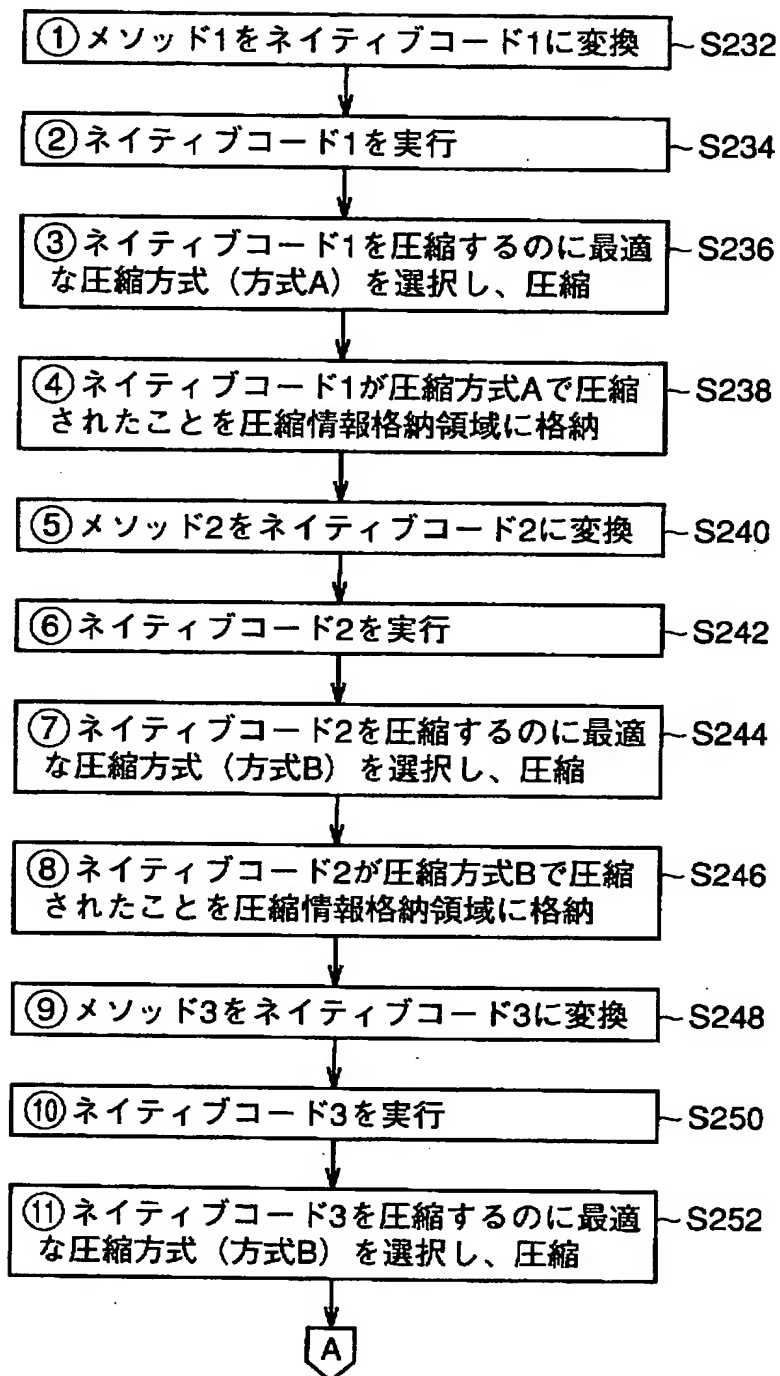
【図 17】



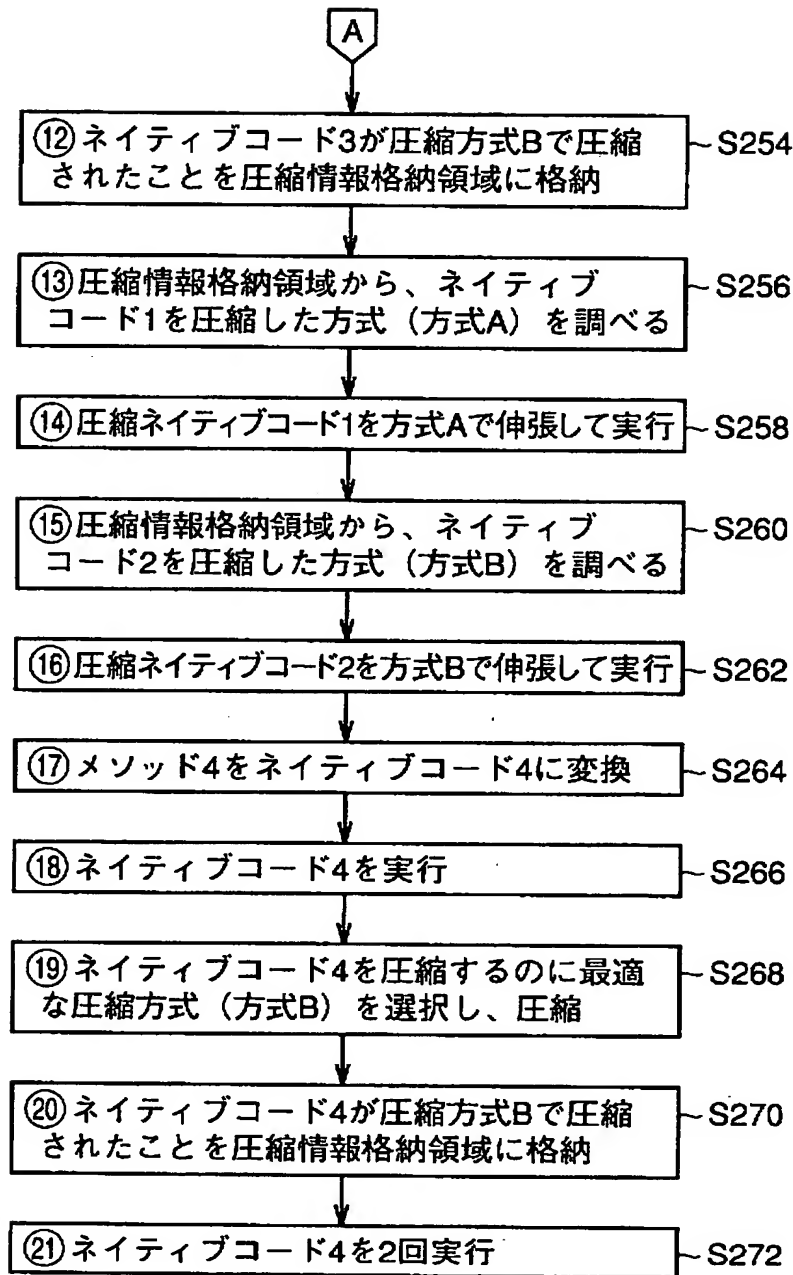
【図 1 8】



【図 1 9】

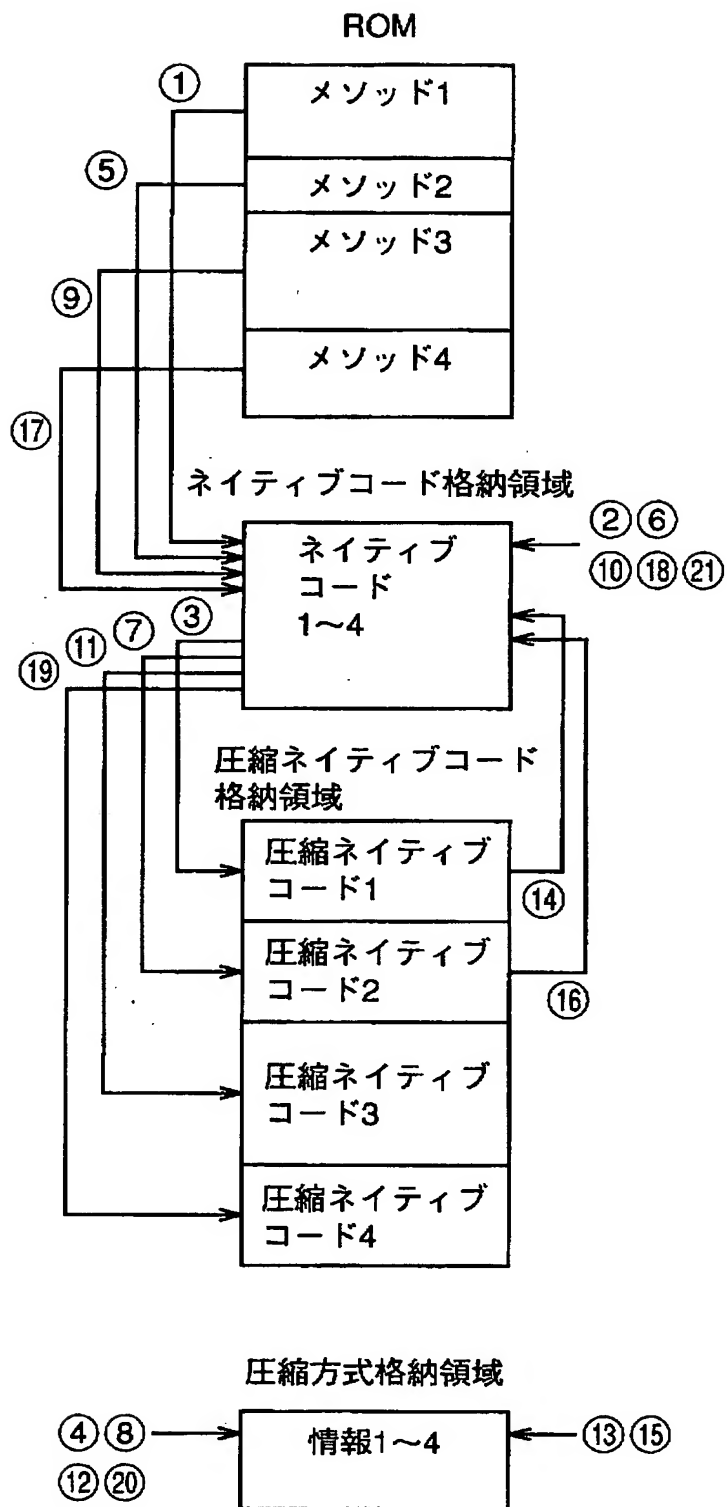


【図 2 0】

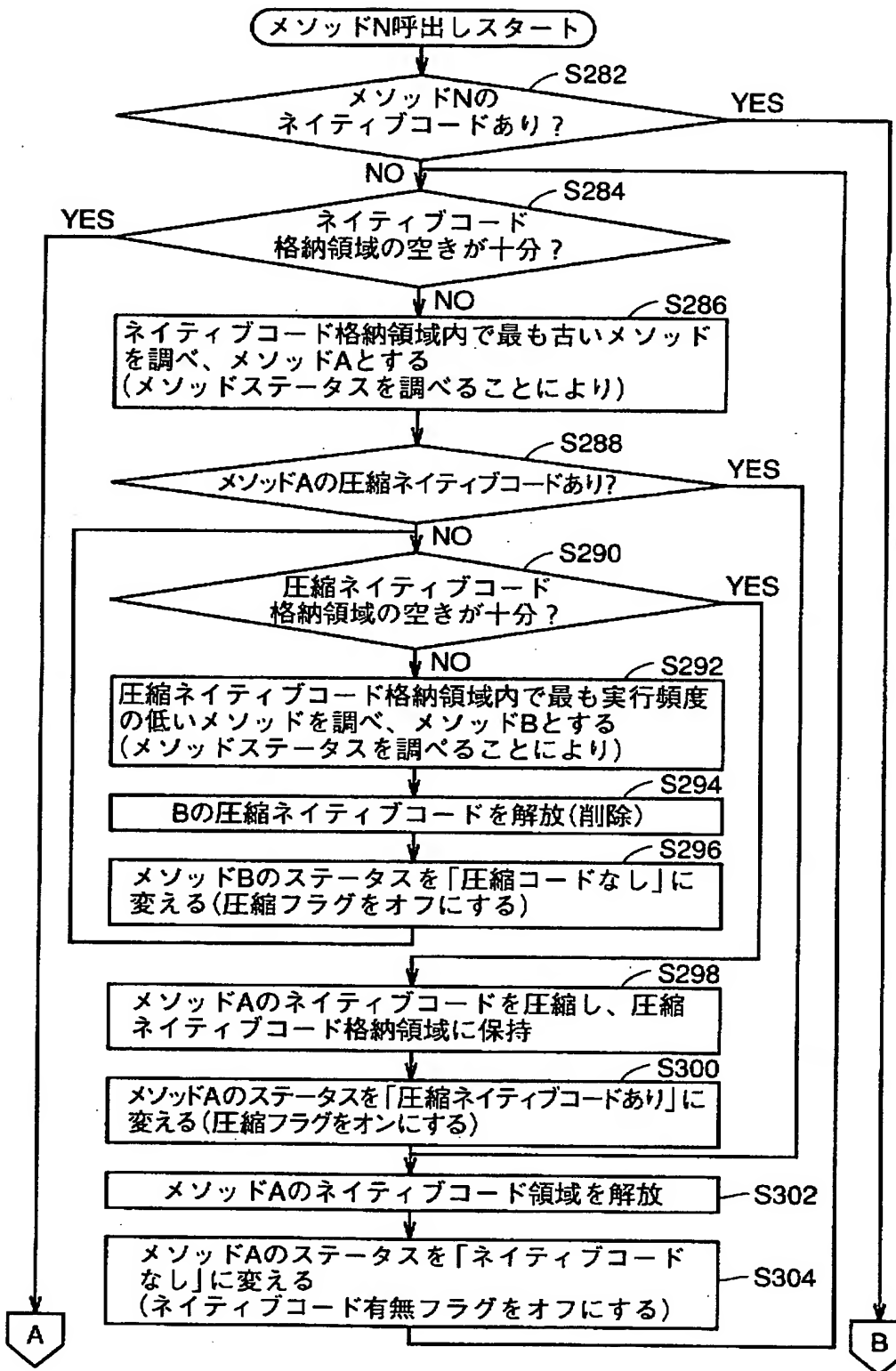




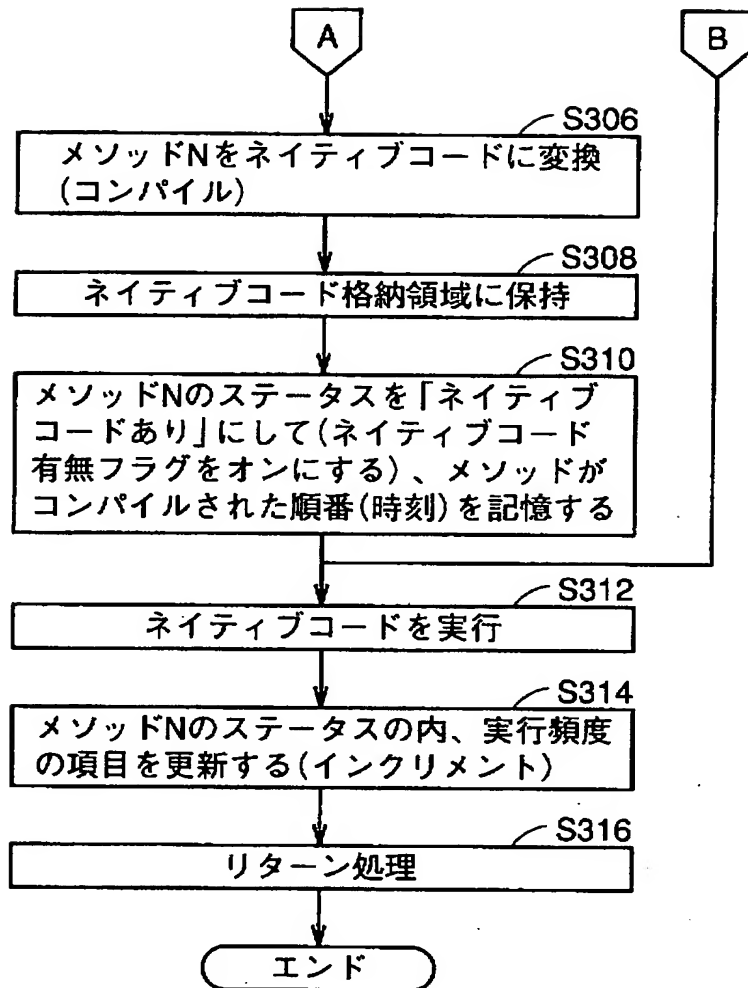
【図 2 1】



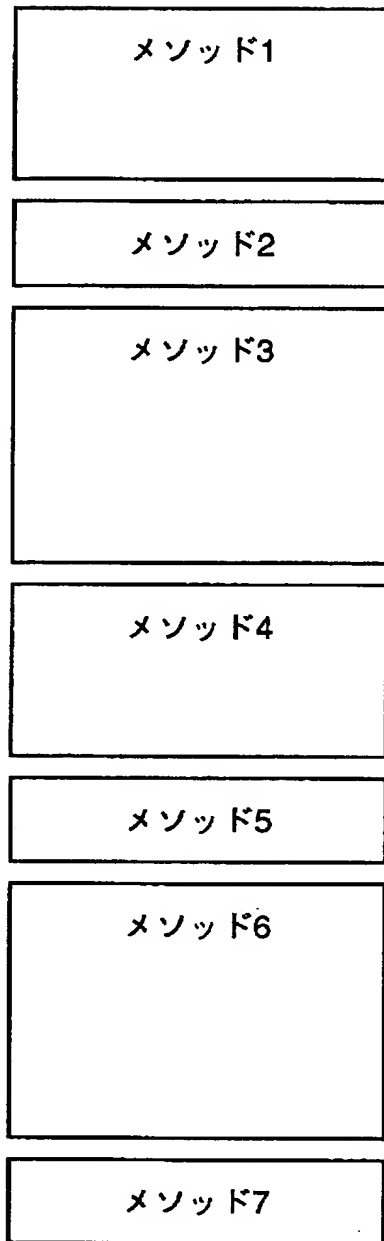
【図 22】



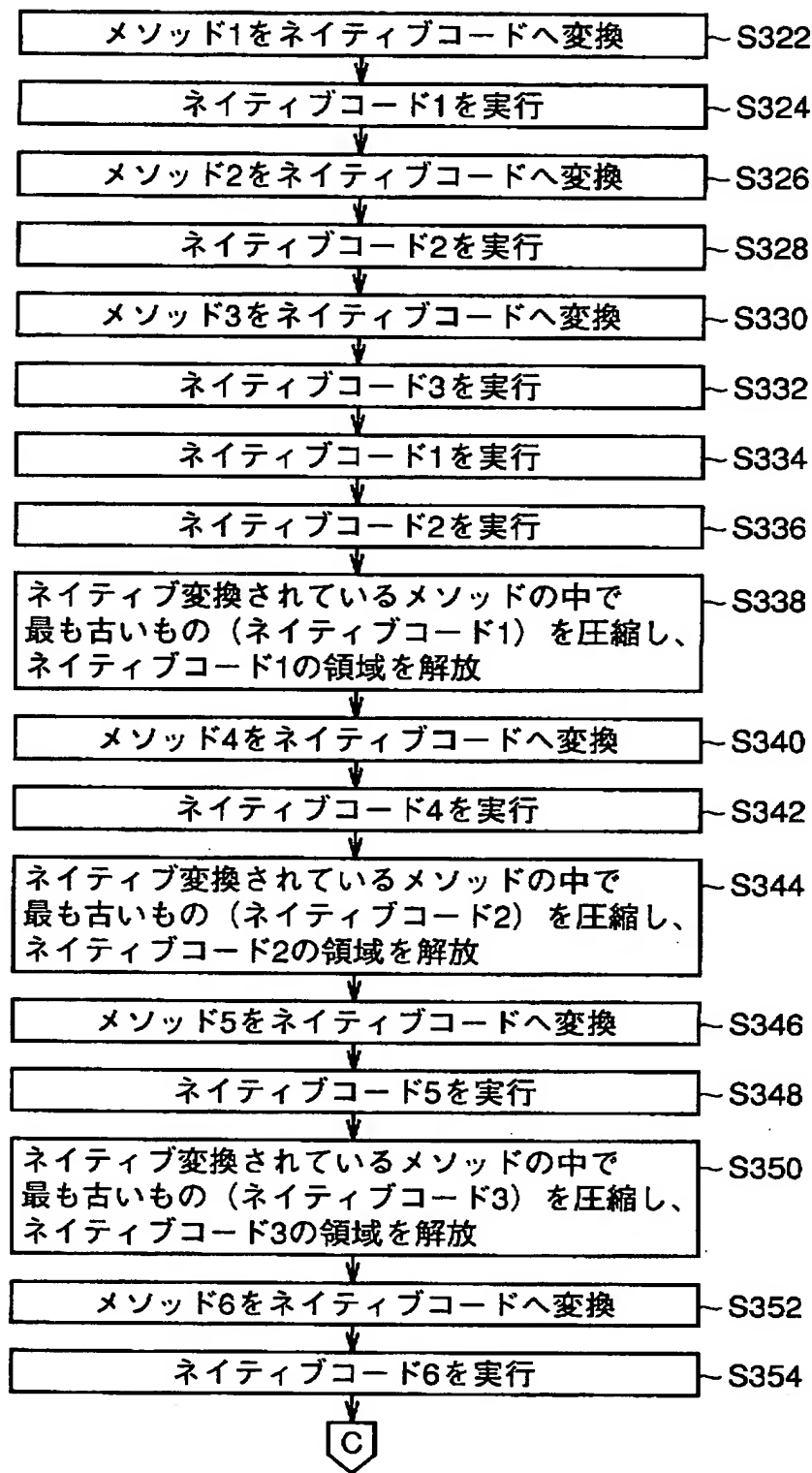
【図 2 3】



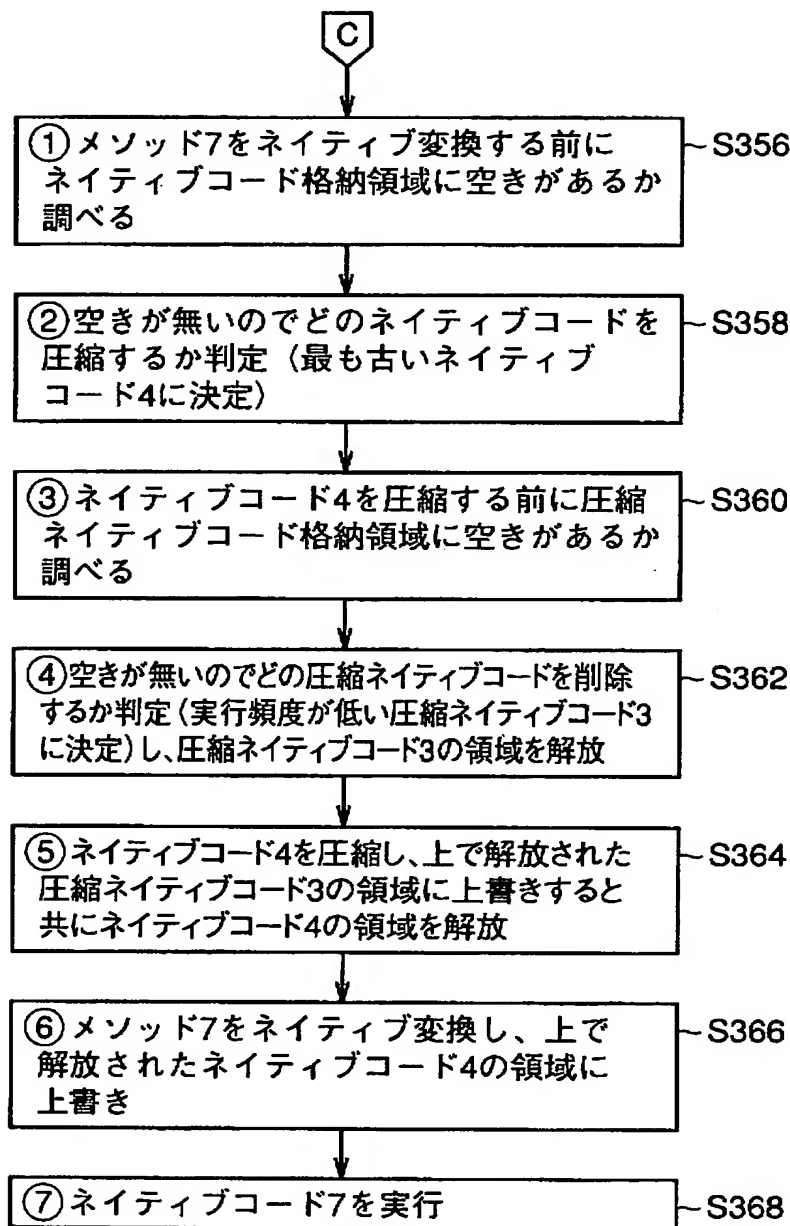
【図 2 4】



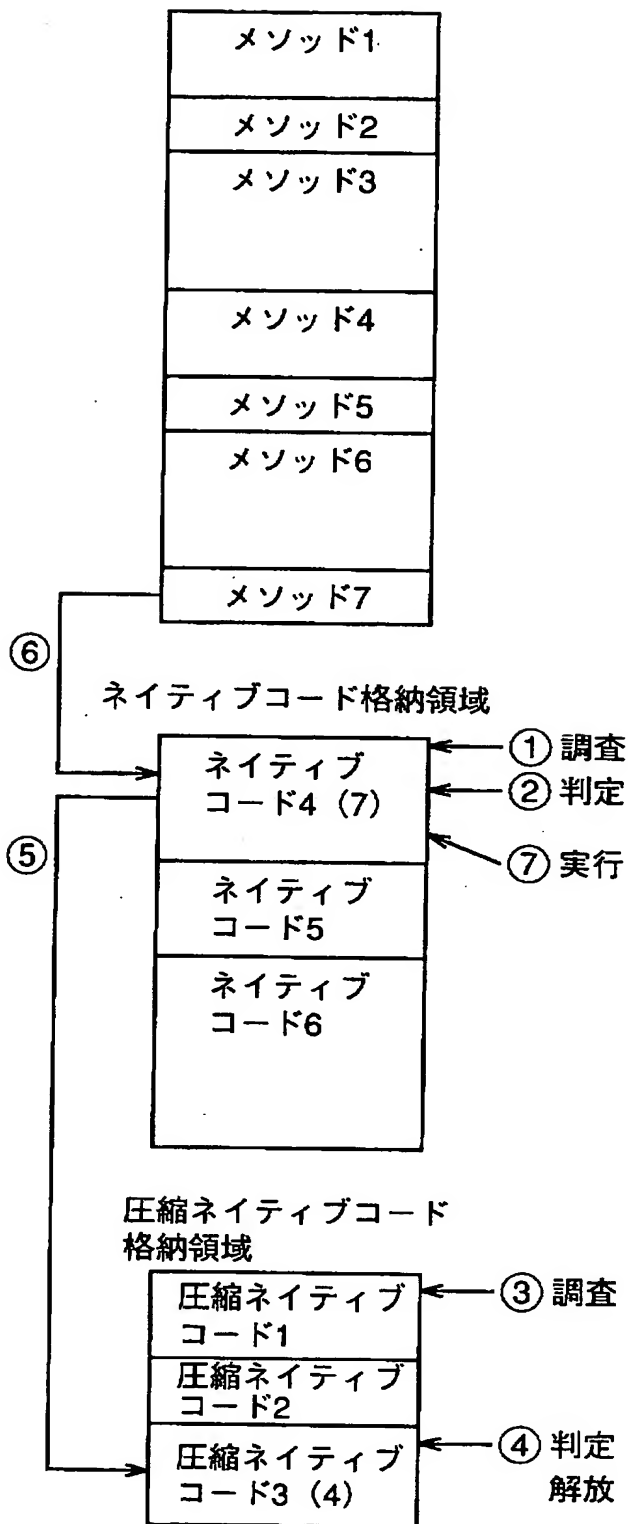
【図 2 5】



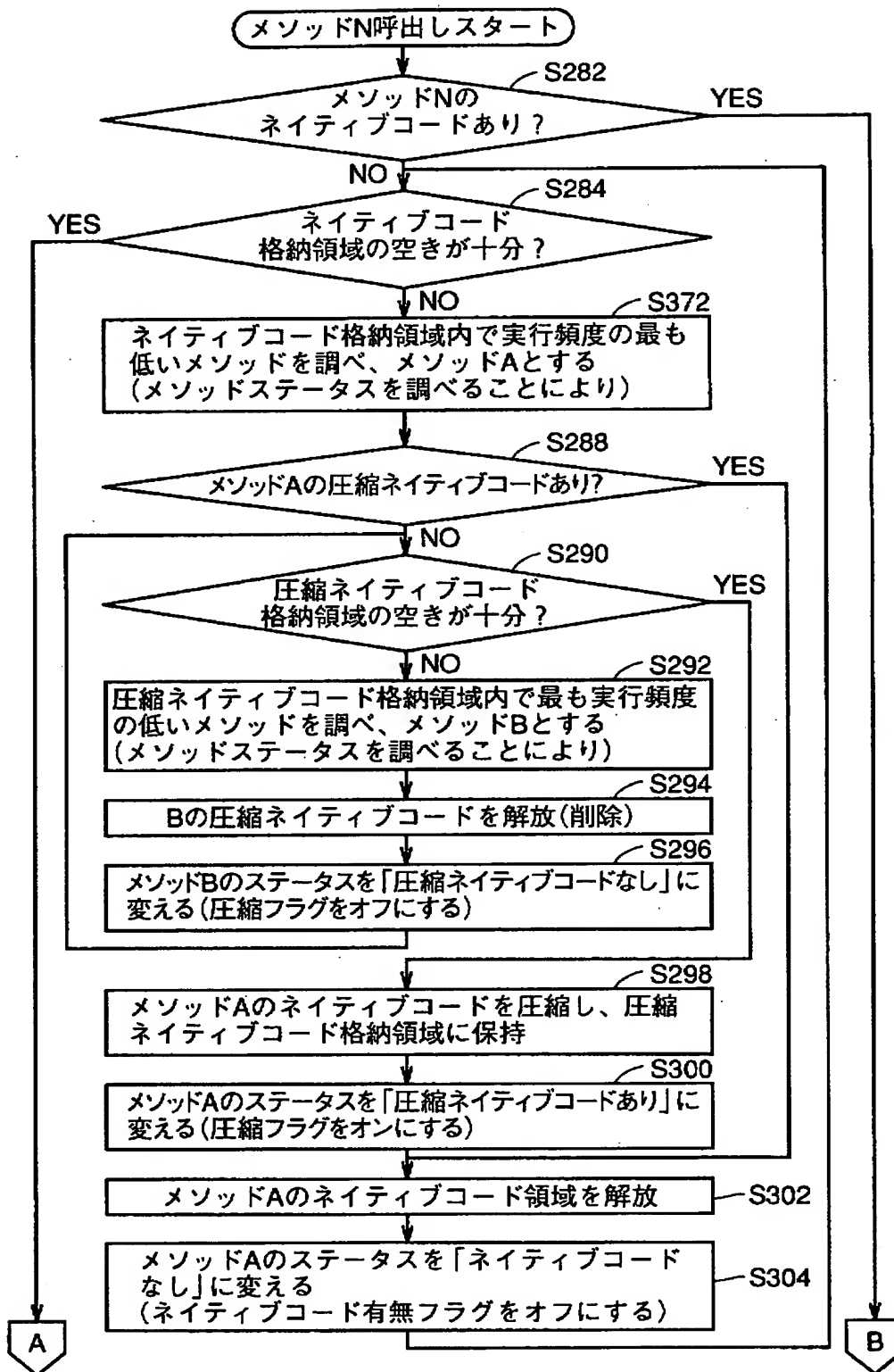
【図 2 6】



【図 27】



【図 28】

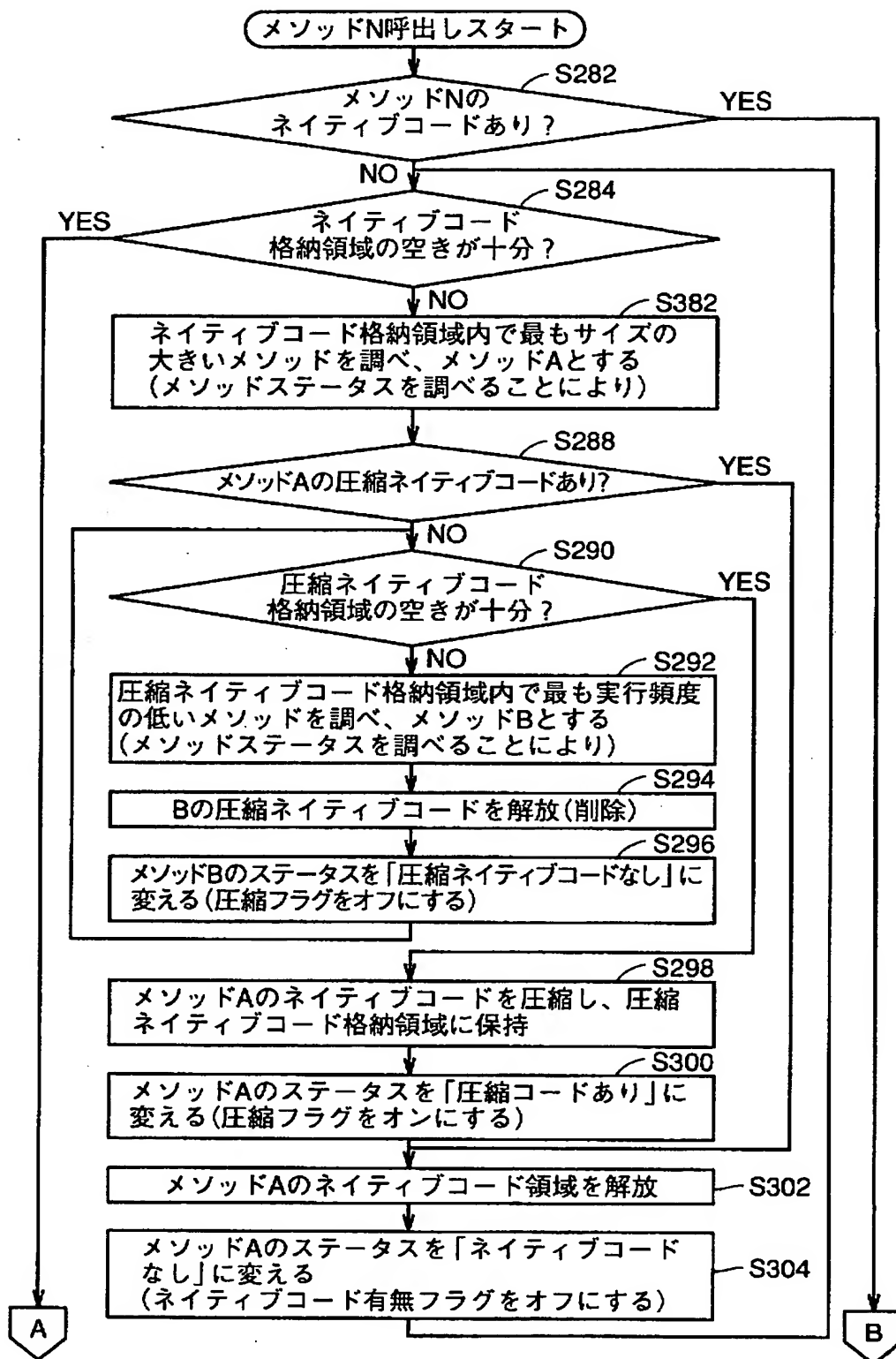




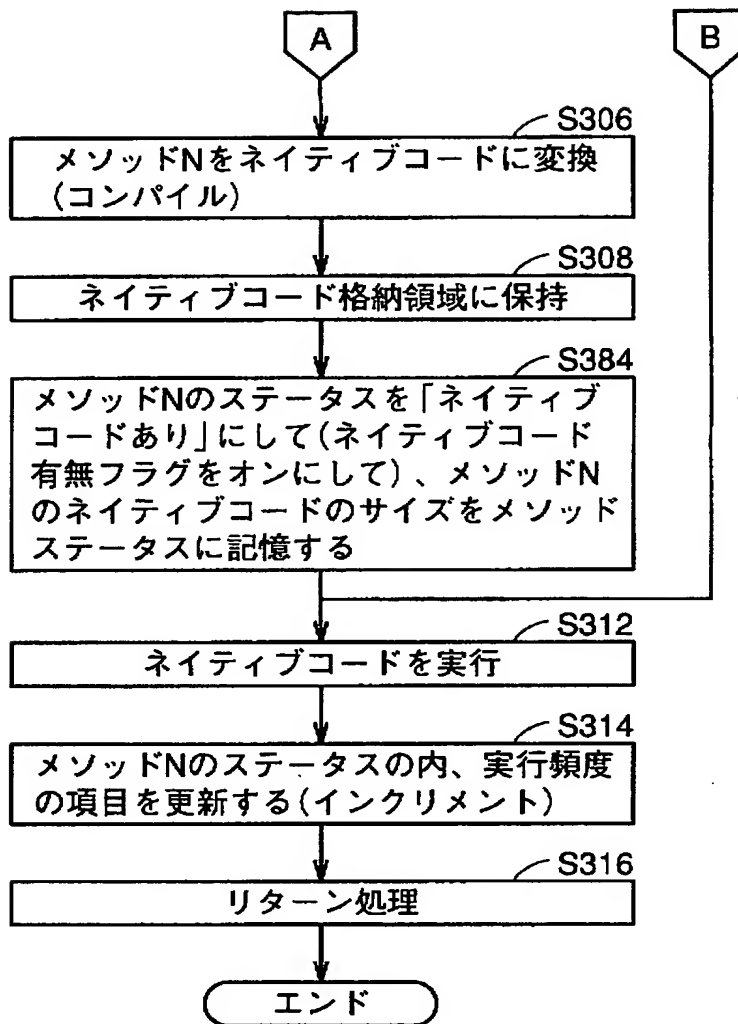
【図 2 9】



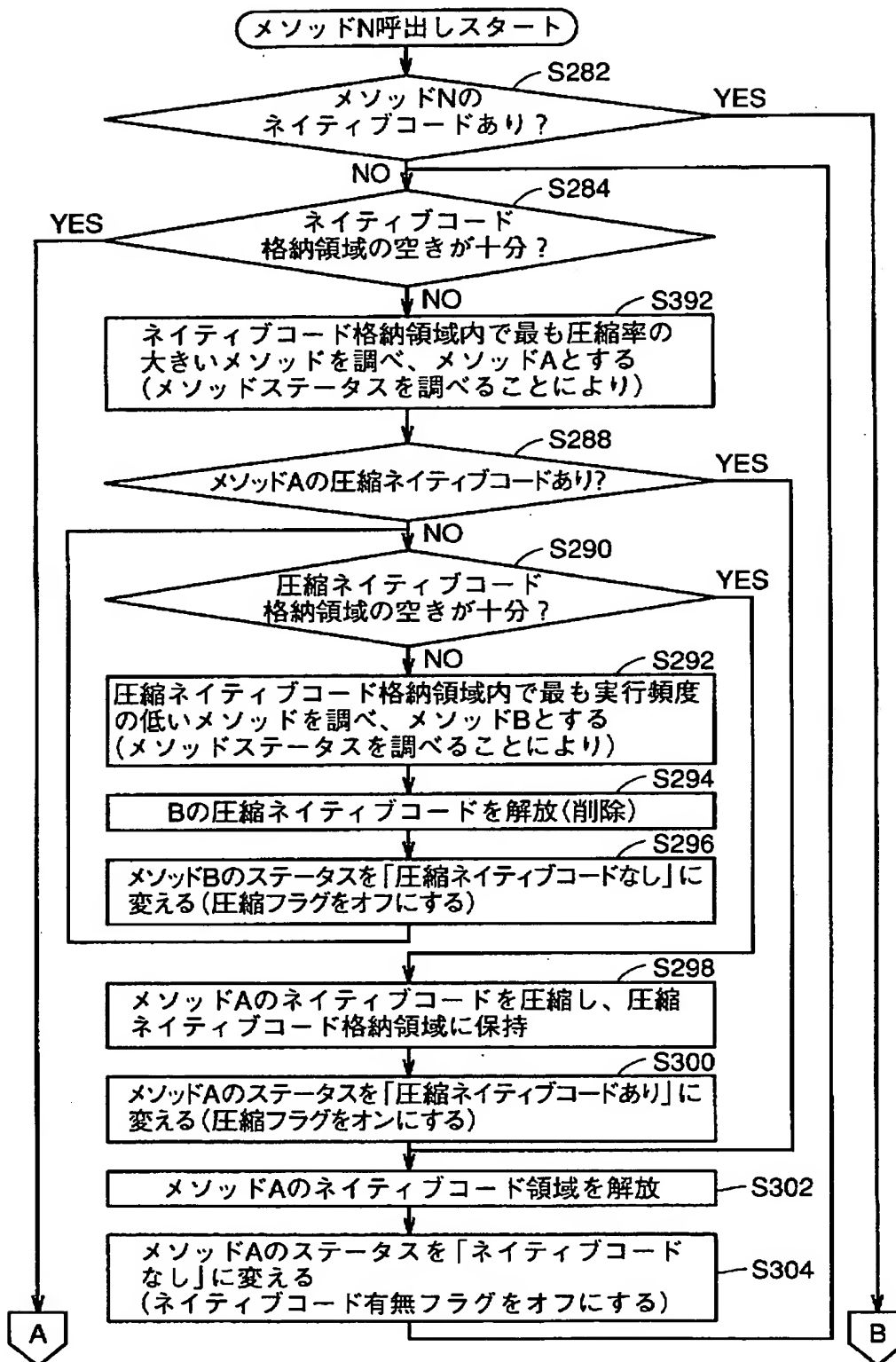
【図30】



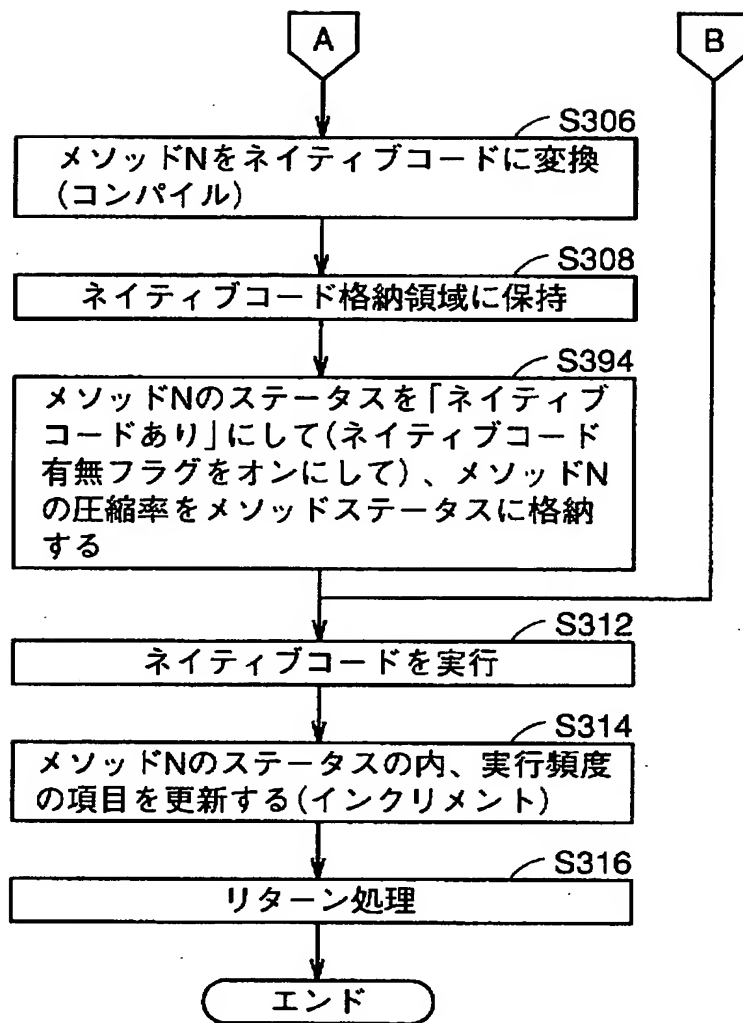
【図 3 1】



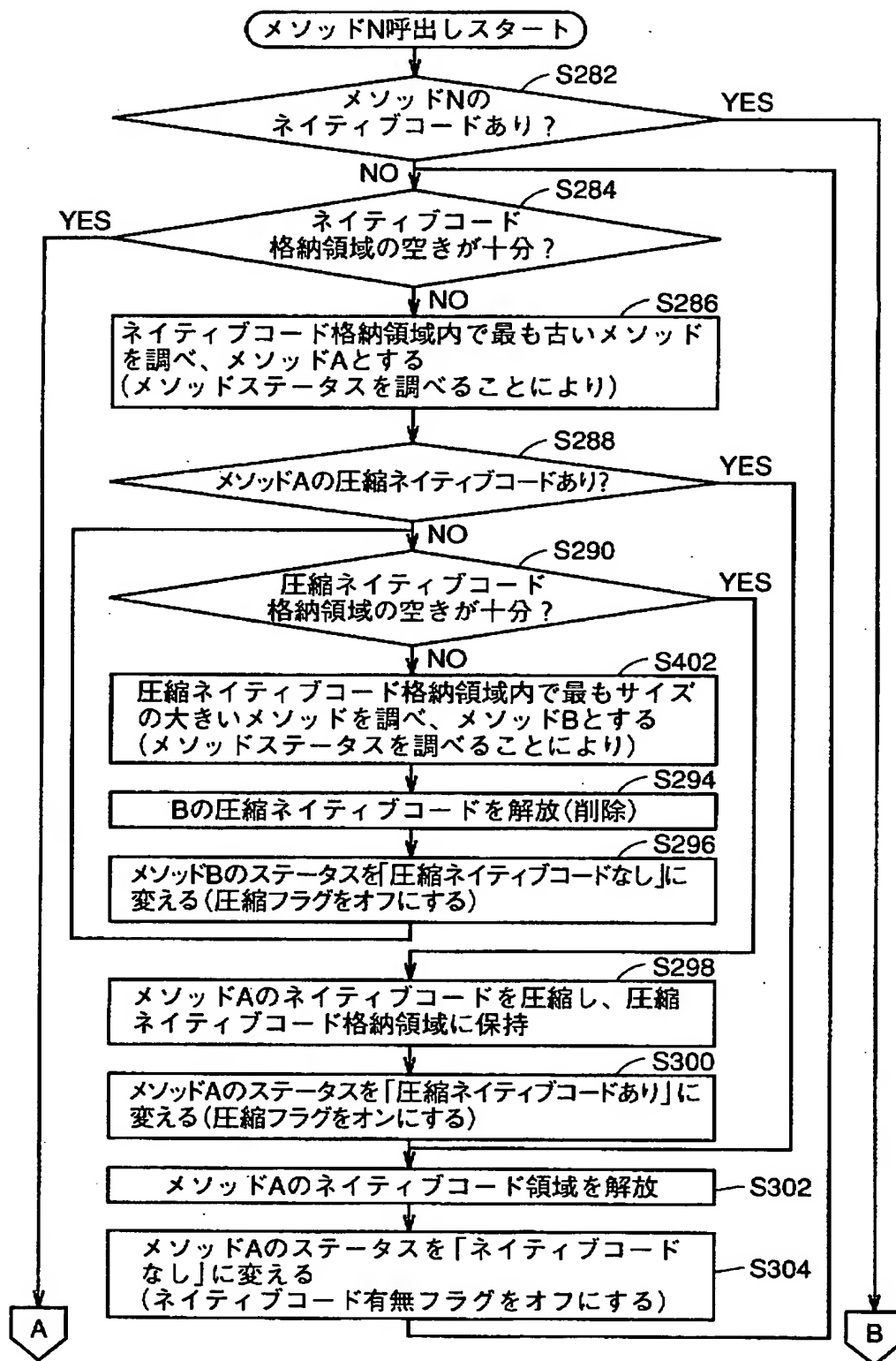
【図 3 2】



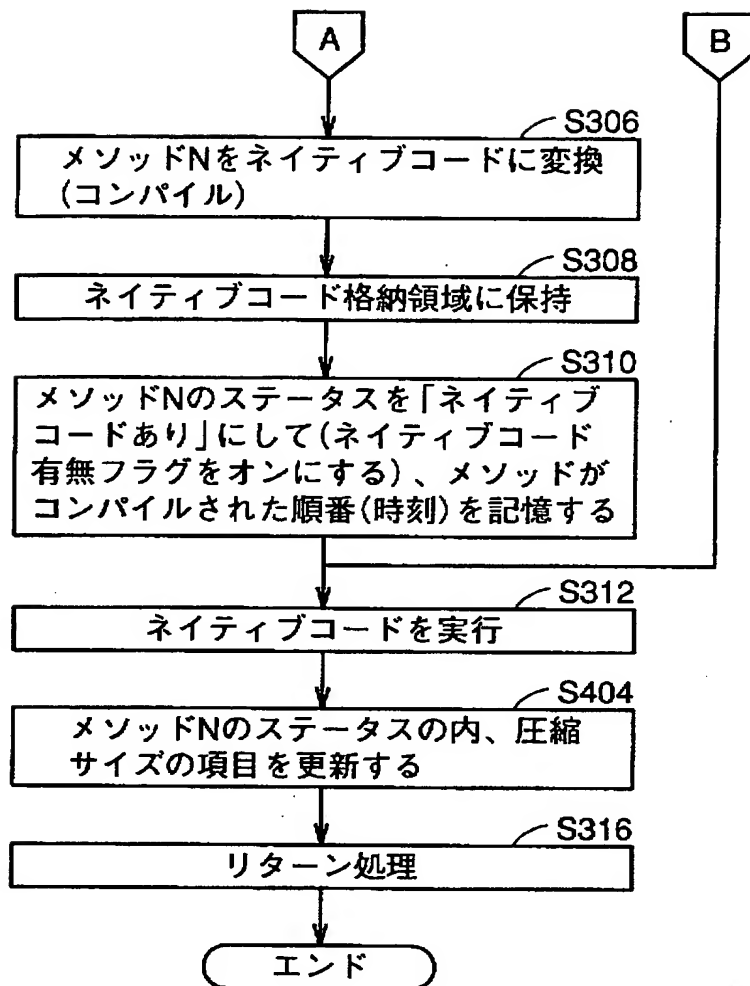
【図 33】



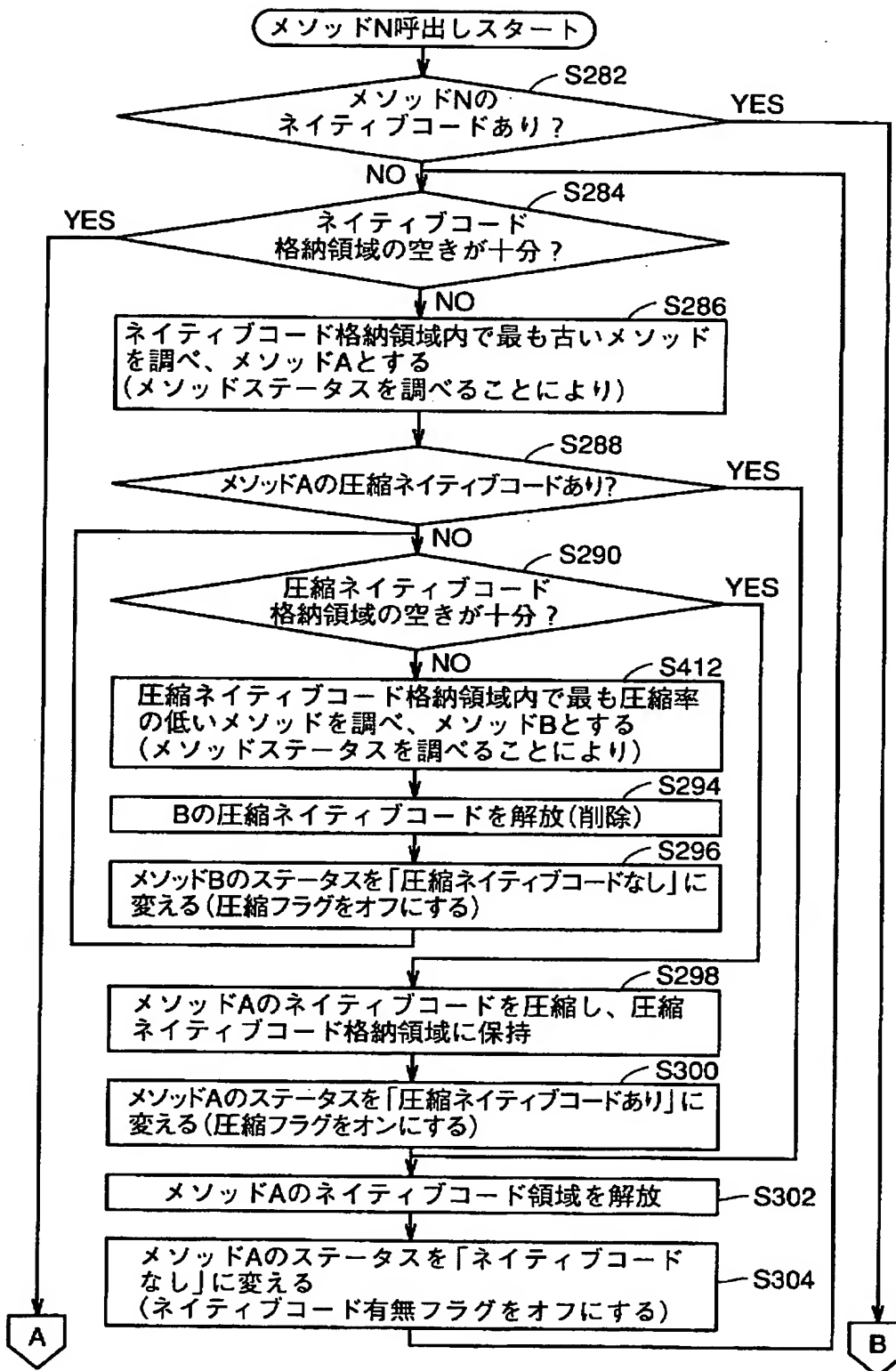
【図 3 4】



【図35】

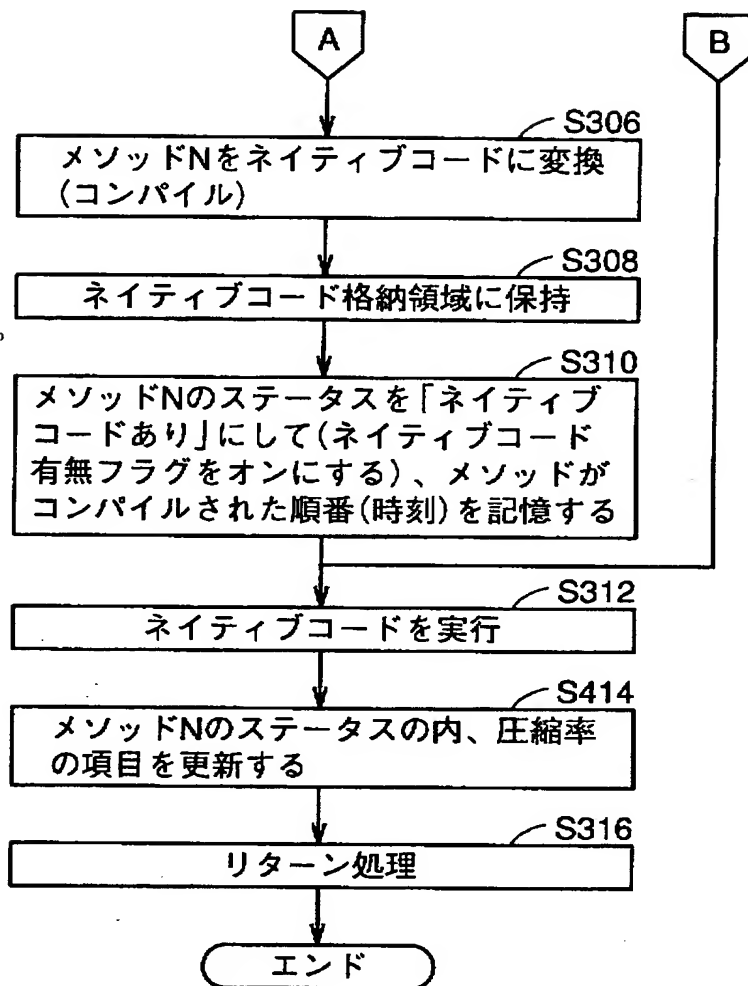


【図 36】

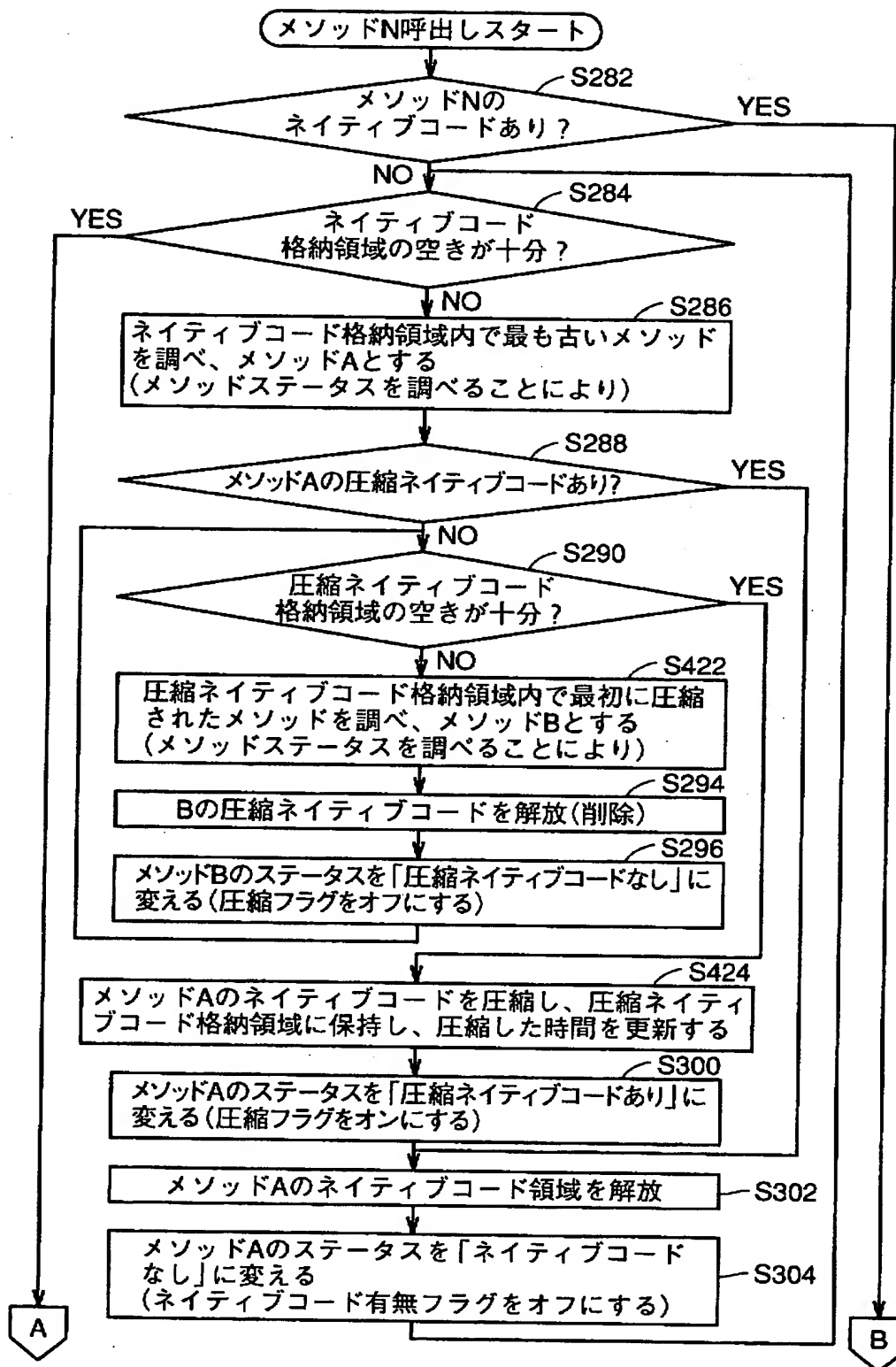




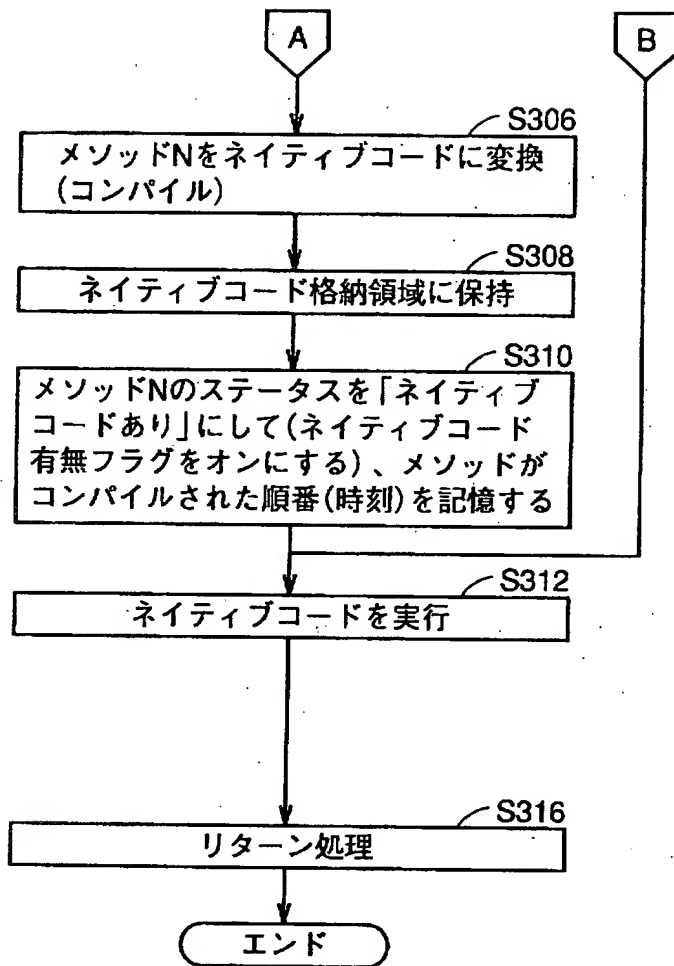
【図 3 7】



【図 3 8】



【図39】



【書類名】 要約書

【要約】

【課題】 必要なメモリの容量が小さいプログラム実行装置を提供する。

【解決手段】 上位のモジュールよりあるメソッドの呼出しがあった場合には、呼出されたメソッドのバイトコードが、伸張されており、伸張済みバイトコード格納領域に格納されているか否かを調べる（S2）。バイトコードが伸張されていなければ（S2でNO）、圧縮バイトコード格納領域に格納されたバイトコードが伸張され（S4）、伸張後のバイトコードが伸張済みバイトコード格納領域に格納される（S6）。このメソッドの圧縮バイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（S8）。インタプリタを用いて、伸張済みのバイトコードが1命令ずつ解釈されながら実行される（S10）。その後、呼出し側のモジュールへ復帰するための処理を行なう（S12）。

【選択図】 図5

特2001-124824

出 願 人 履 歴 情 報

識別番号 [000006013]

1. 変更年月日 1990年 8月24日

[変更理由] 新規登録

住 所 東京都千代田区丸の内2丁目2番3号

氏 名 三菱電機株式会社